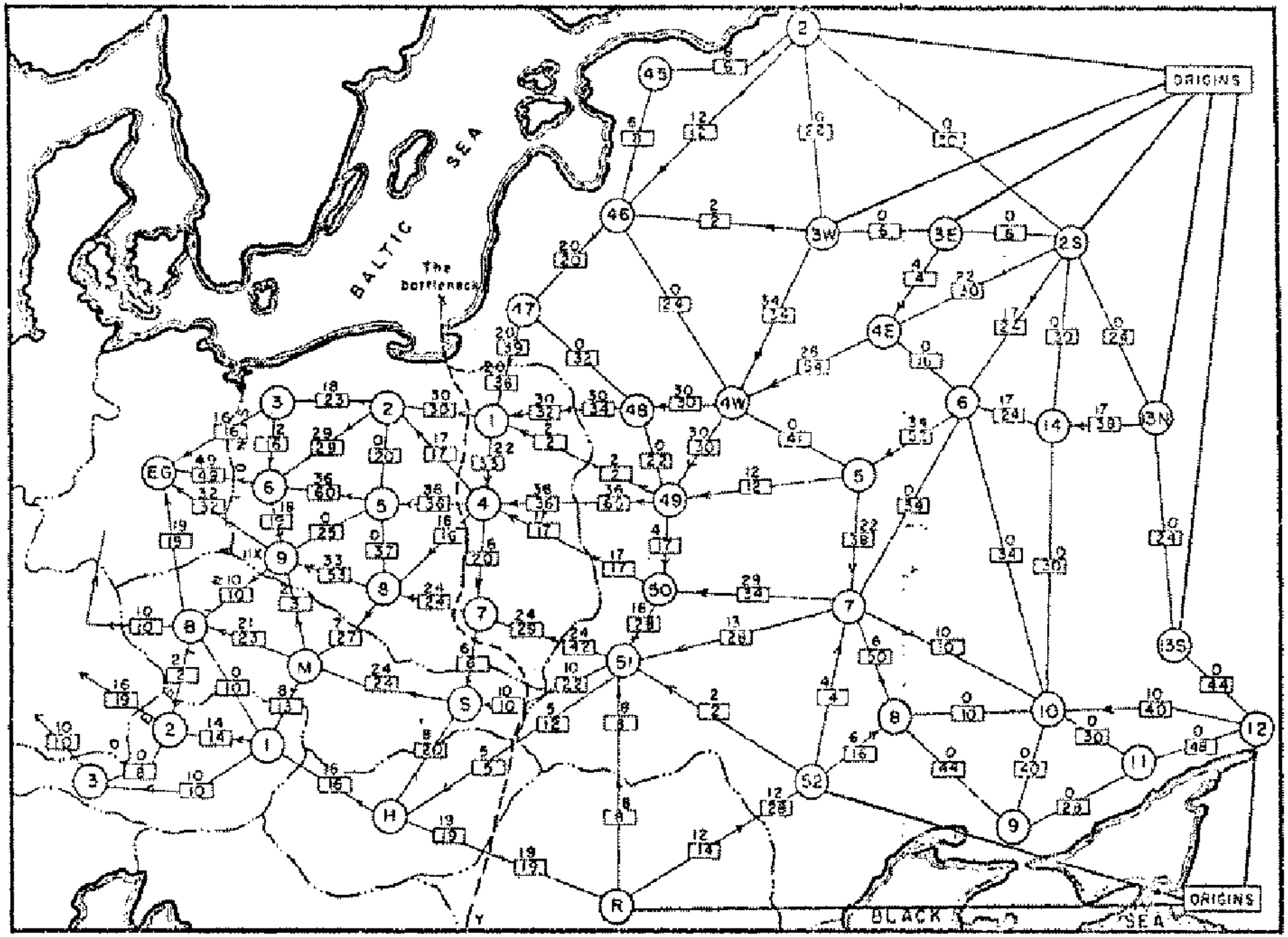
CSE 421

Max Flow Min Cut Problem

Yin Tat Lee

Soviet Rail Network



“Unclassified” on May 21, 1999.

Network Flow Applications

Max flow and min cut.

- Two very rich algorithmic problems.
- Cornerstone problems in combinatorial optimization.
- Beautiful mathematical duality.

Nontrivial applications / reductions.

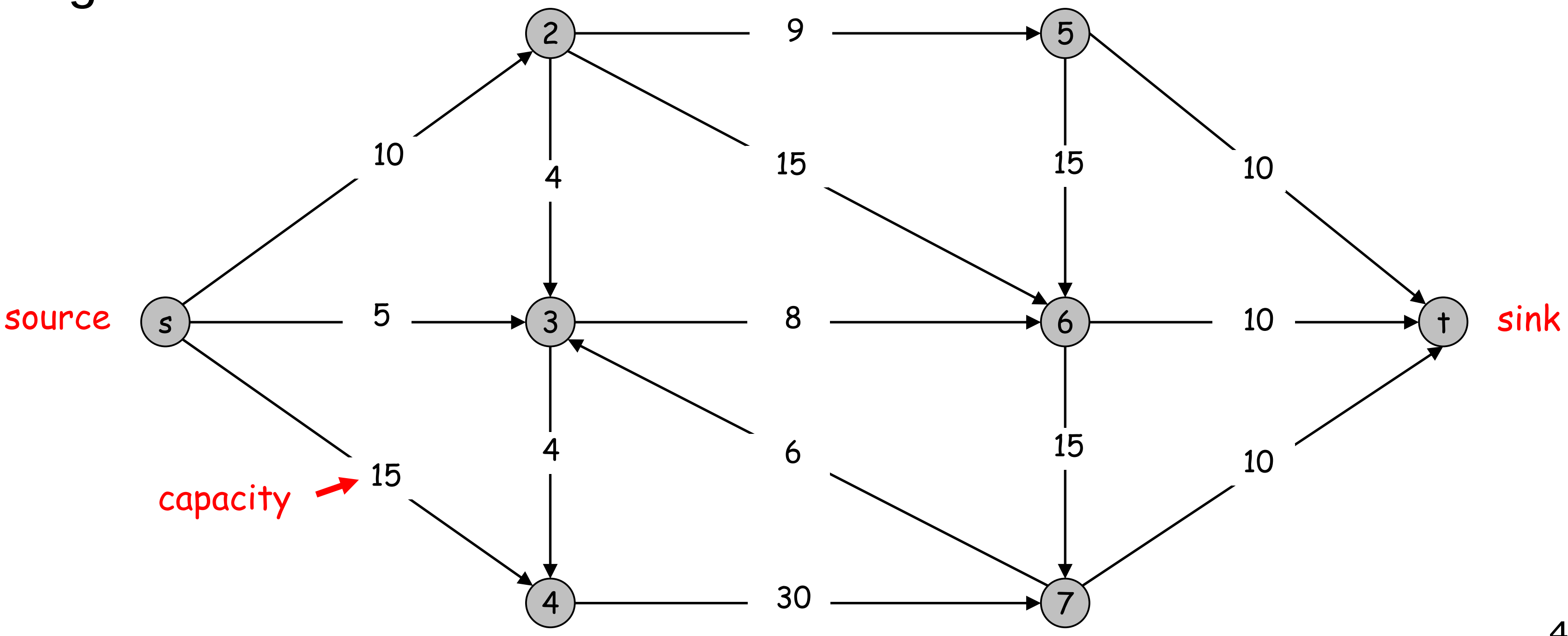
- Data mining.
- Project selection.
- Airline scheduling.
- Bipartite matching.
- Baseball elimination.
- Image segmentation.
- Network connectivity.

Minimum s-t Cut Problem

Given a directed graph G = (V, E) = directed graph and two distinguished nodes: s = source, t = sink.

Suppose each directed edge e has a nonnegative capacity $c(e)$

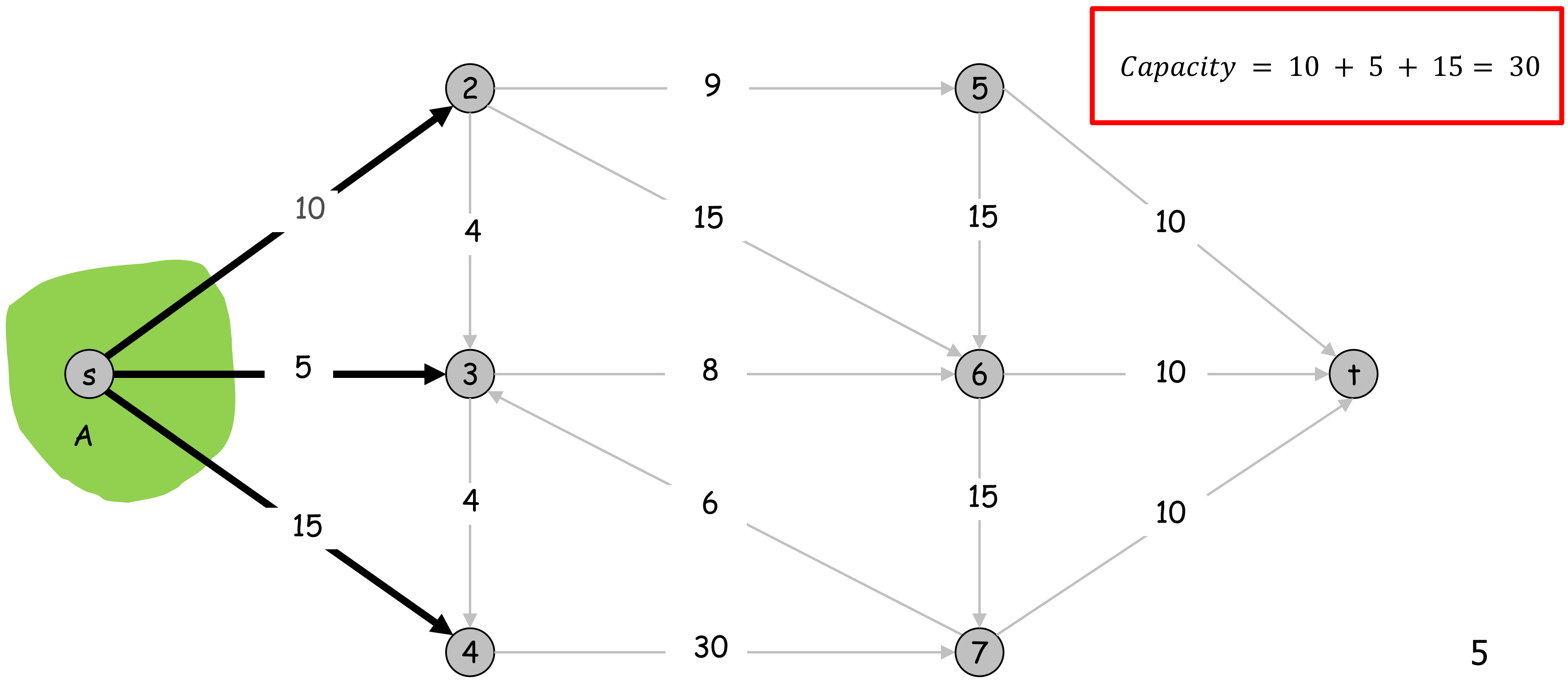
Goal: Find a cut separating s, t that cuts the minimum capacity of edges.



s-t cuts

Def. An s-t cut is a partition (A, B) of V with s $\in$ A and t $\in$ B.

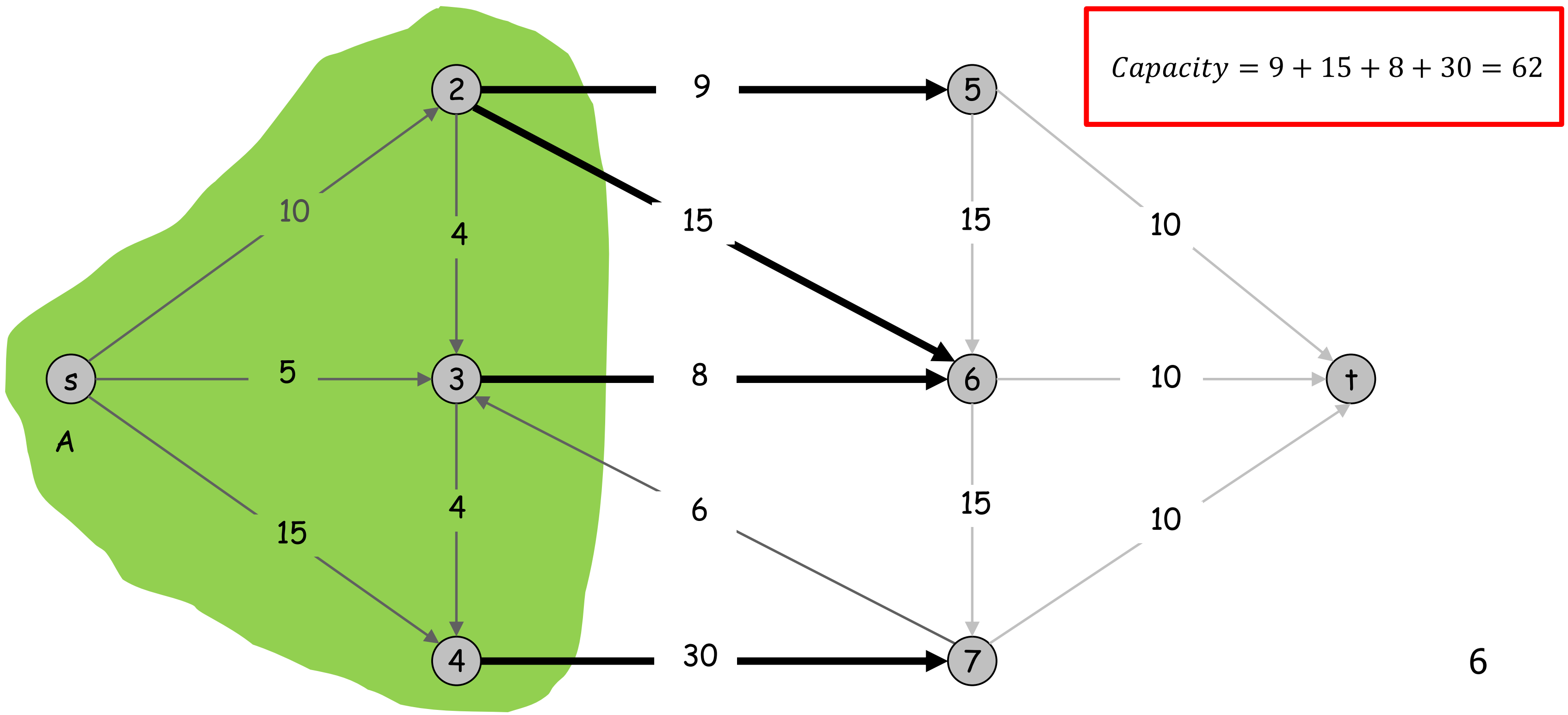
Def. The capacity of a cut (A, B): $cap(A, B) = \sum_{(u,v):u\in A, v\in B} c(u, v)$



s-t cuts

Def. An s-t cut is a partition (A, B) of V with s ∈ A and t ∈ B.

Def. The capacity of a cut (A, B): $cap(A,B) = \sum_{(u,v):u\in A, v\in B} c(u,v)$

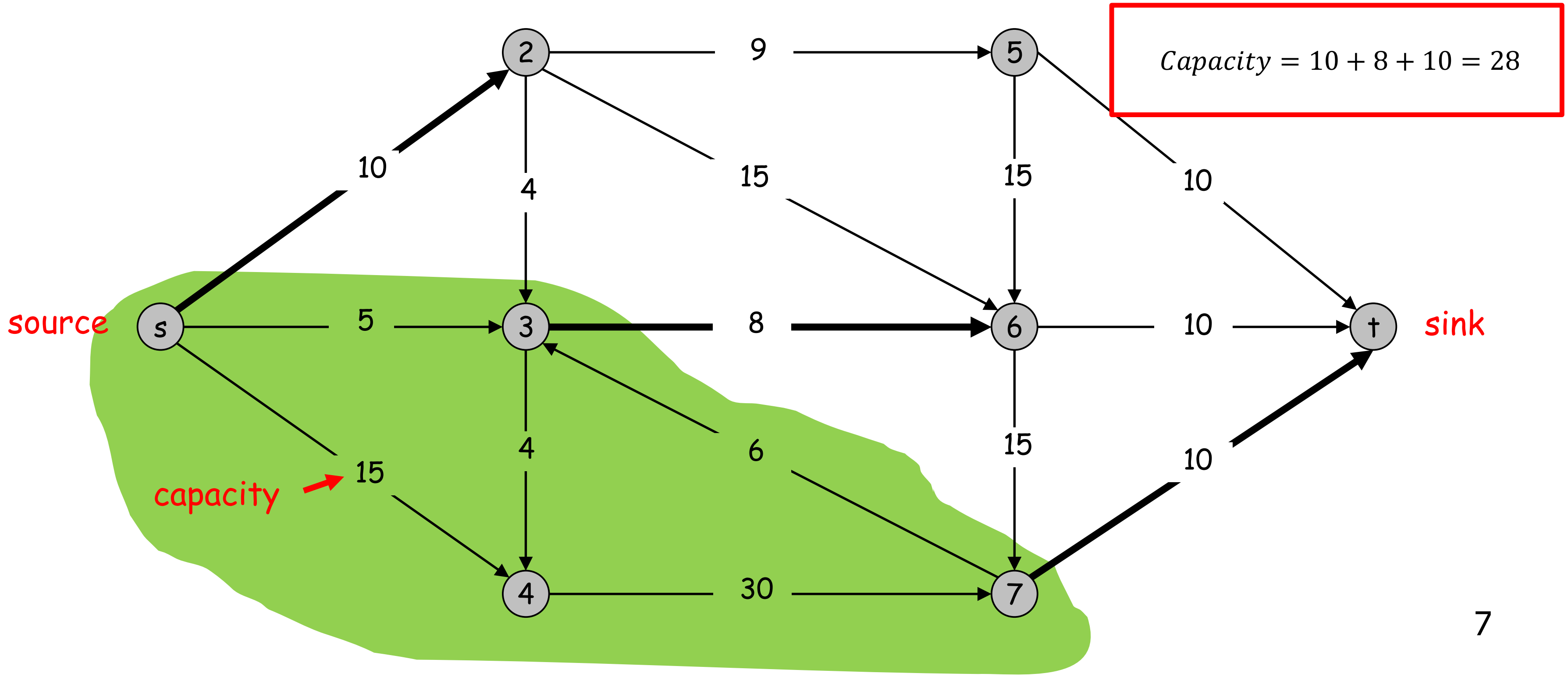


Minimum s-t Cut Problem

Given a directed graph G = (V, E) = directed graph and two distinguished nodes: s = source, t = sink.

Suppose each directed edge e has a nonnegative capacity $c(e)$

Goal: Find a s-t cut of minimum capacity

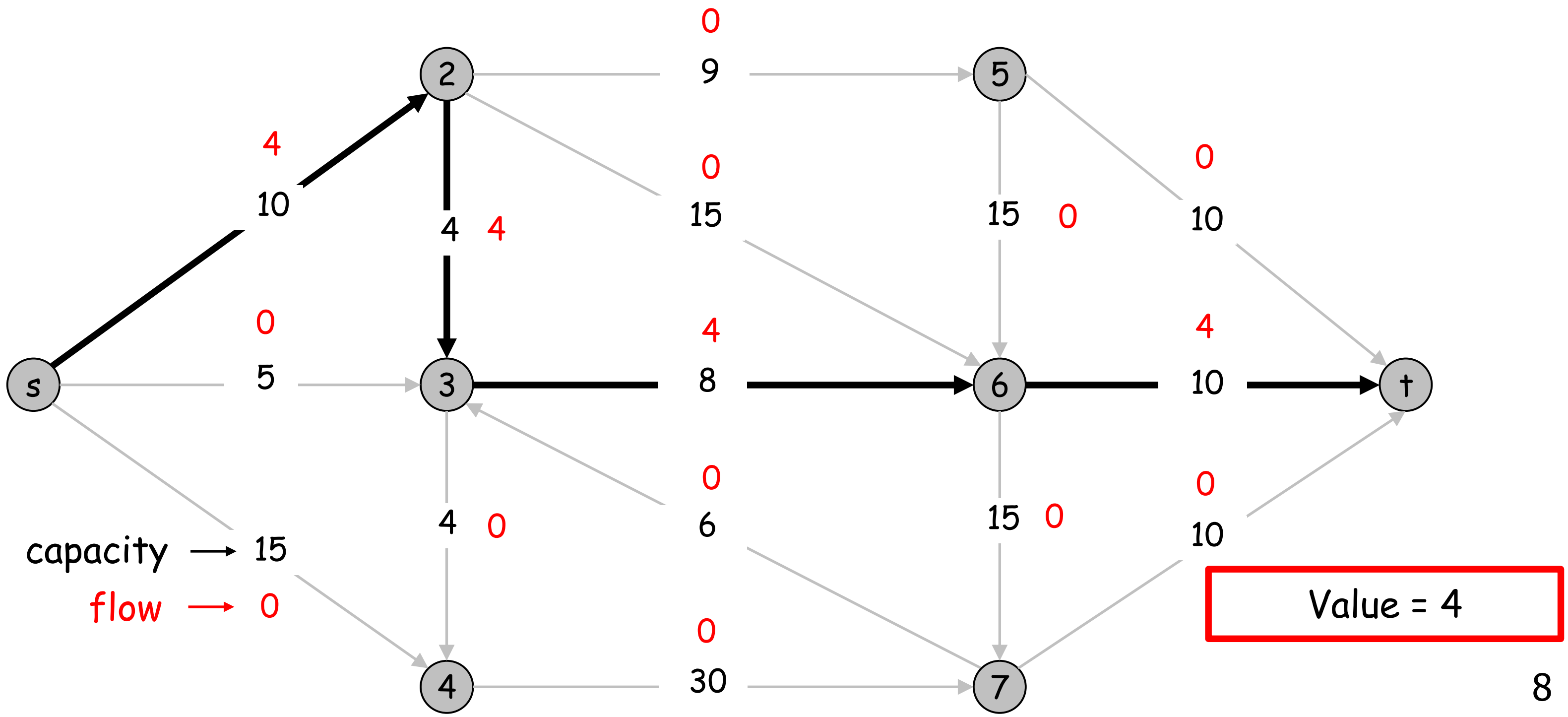


s-t Flows

Def. An s-t flow is a function that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

Def. The value of a flow f is: $v(f) = \sum_{e \text{ out of } s} f(e)$

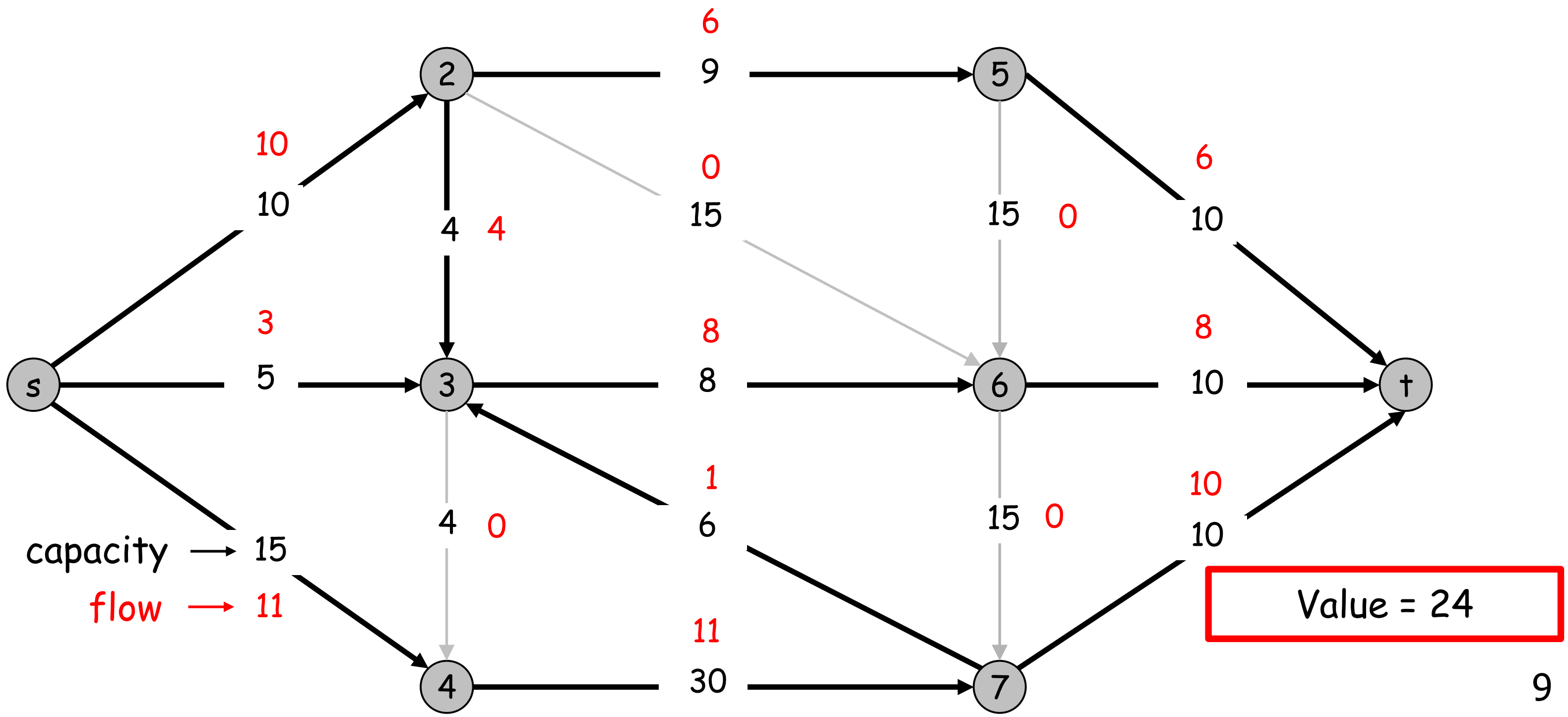


s-t Flows

Def. An s-t flow is a function that satisfies:

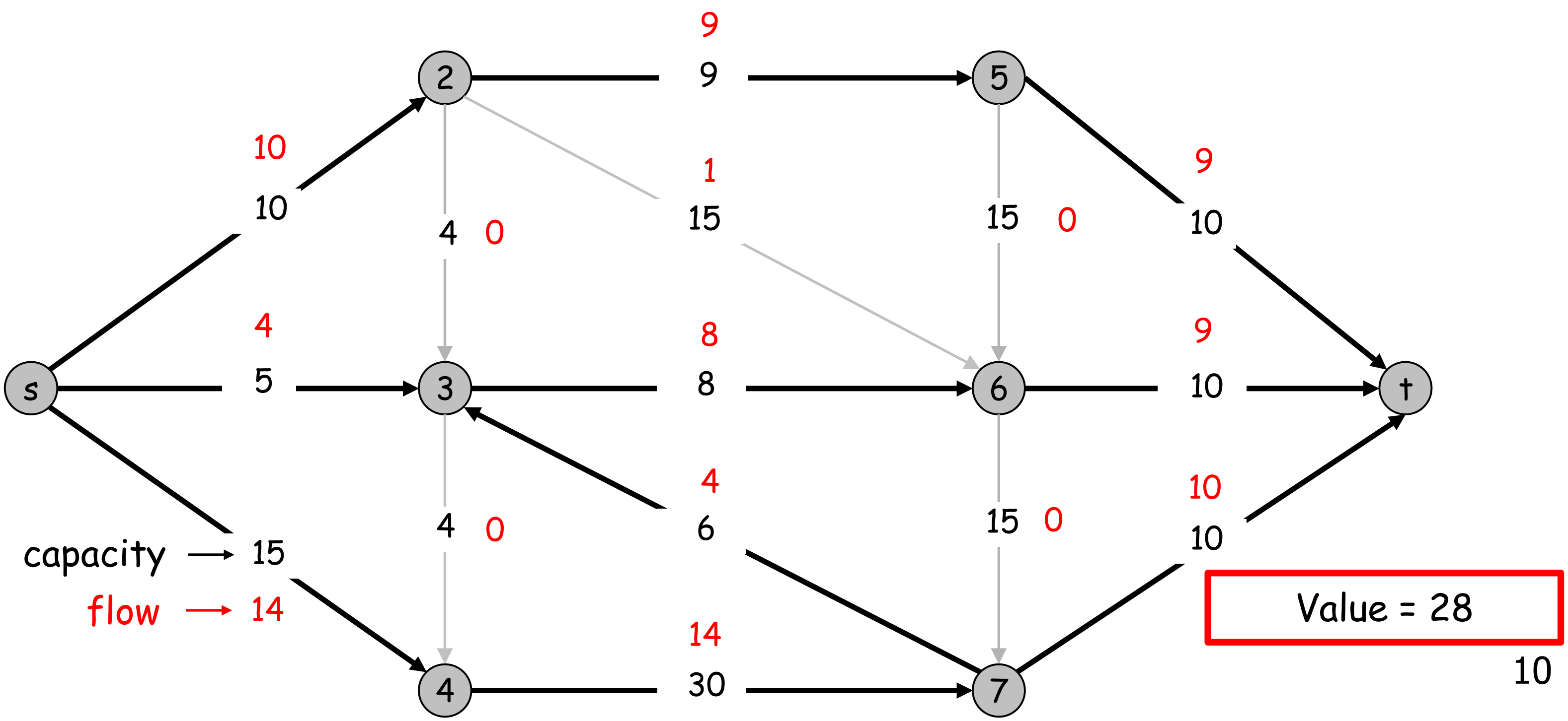
- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

Def. The value of a flow f is: $v(f) = \sum_{e \text{ out of } s} f(e)$



Maximum s-t Flow Problem

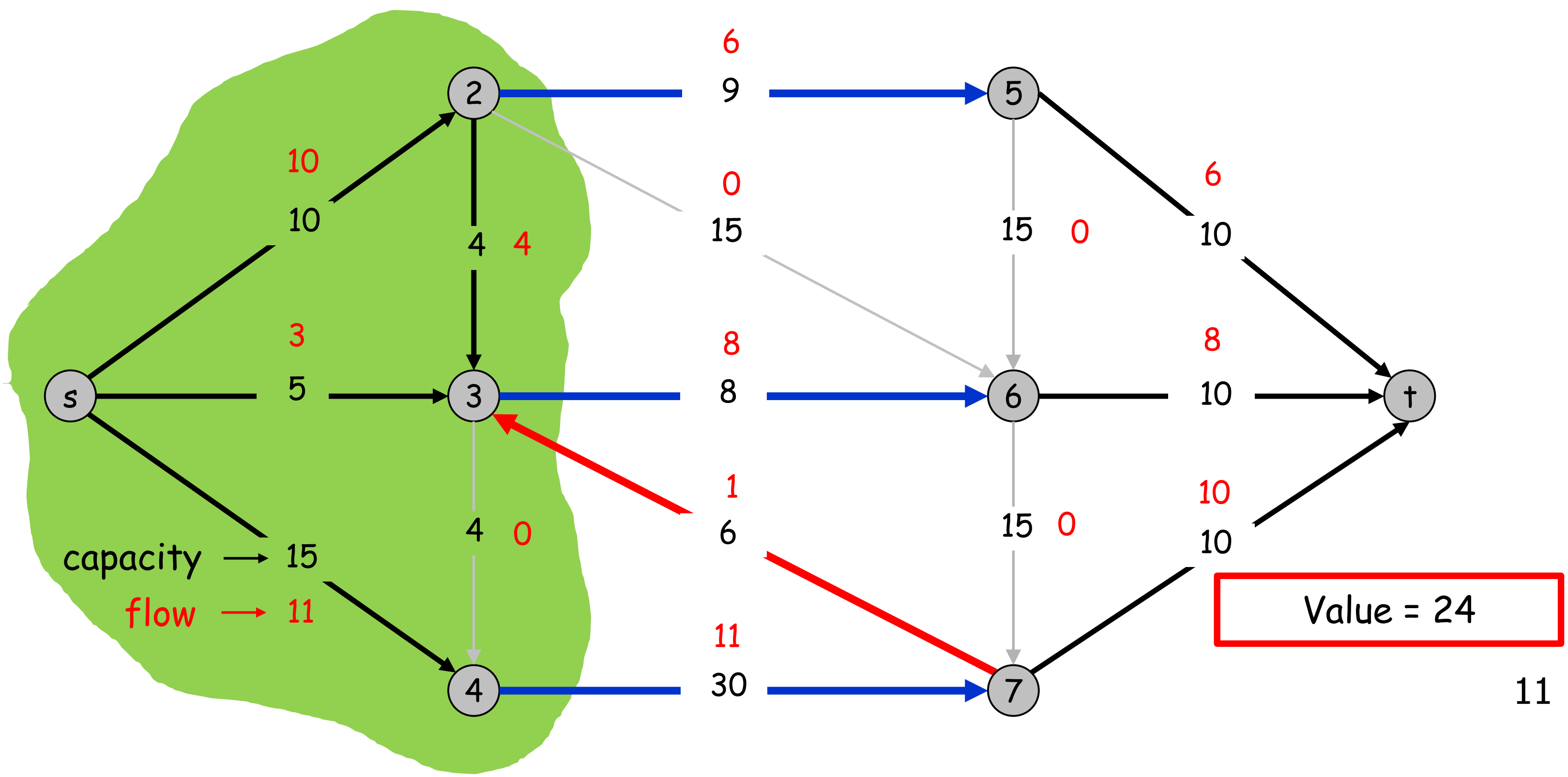
Goal: Find a s-t flow of largest value.



Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then, the net flow sent across the cut is equal to the amount leaving s.

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$



Proof of Flow value Lemma

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then, the net flow sent across the cut is equal to the amount leaving s.

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$

Proof.

$$v(f) = \sum_{e \text{ out of } s} f(e)$$

By conservation of flow, all terms except v=s are 0 →

$$= \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) \right)$$

All contributions due to internal edges cancel out →

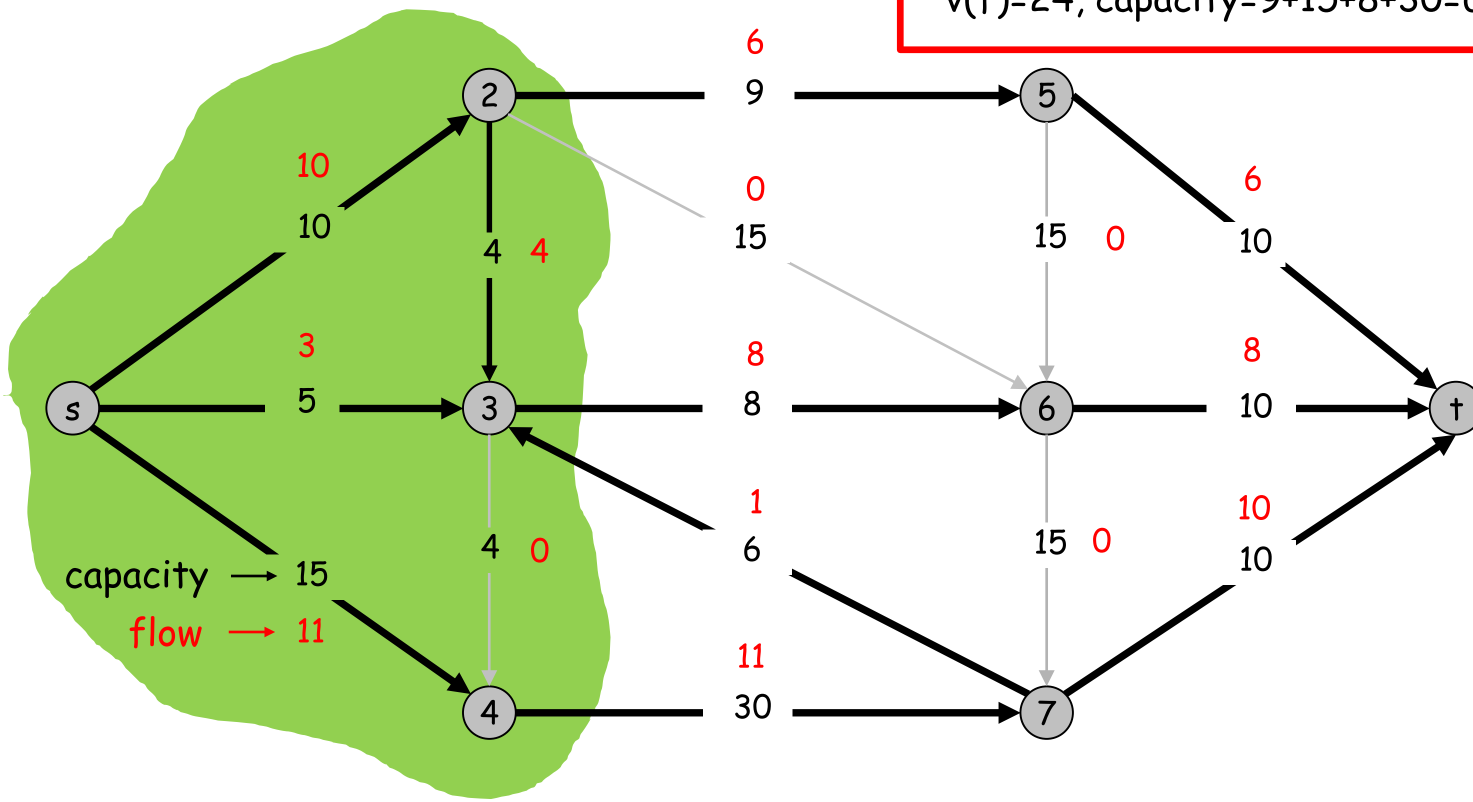
$$= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

Weak Duality of Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then the value of the flow is at most the capacity of the cut.

$$v(f) \leq cap(A, B)$$

v(f)=24, capacity=9+15+8+30=62



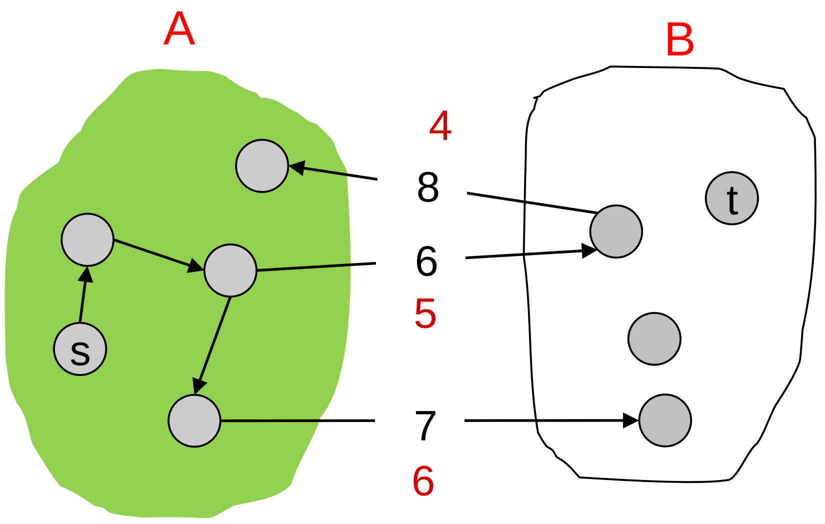
Weak Duality of Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then the value of the flow is at most the capacity of the cut.

$$v(f) \leq cap(A, B)$$

Proof.

$$\begin{aligned} v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\ &\leq \sum_{e \text{ out of } A} f(e) \\ &\leq \sum_{e \text{ out of } A} c(e) = cap(A, B) \end{aligned}$$

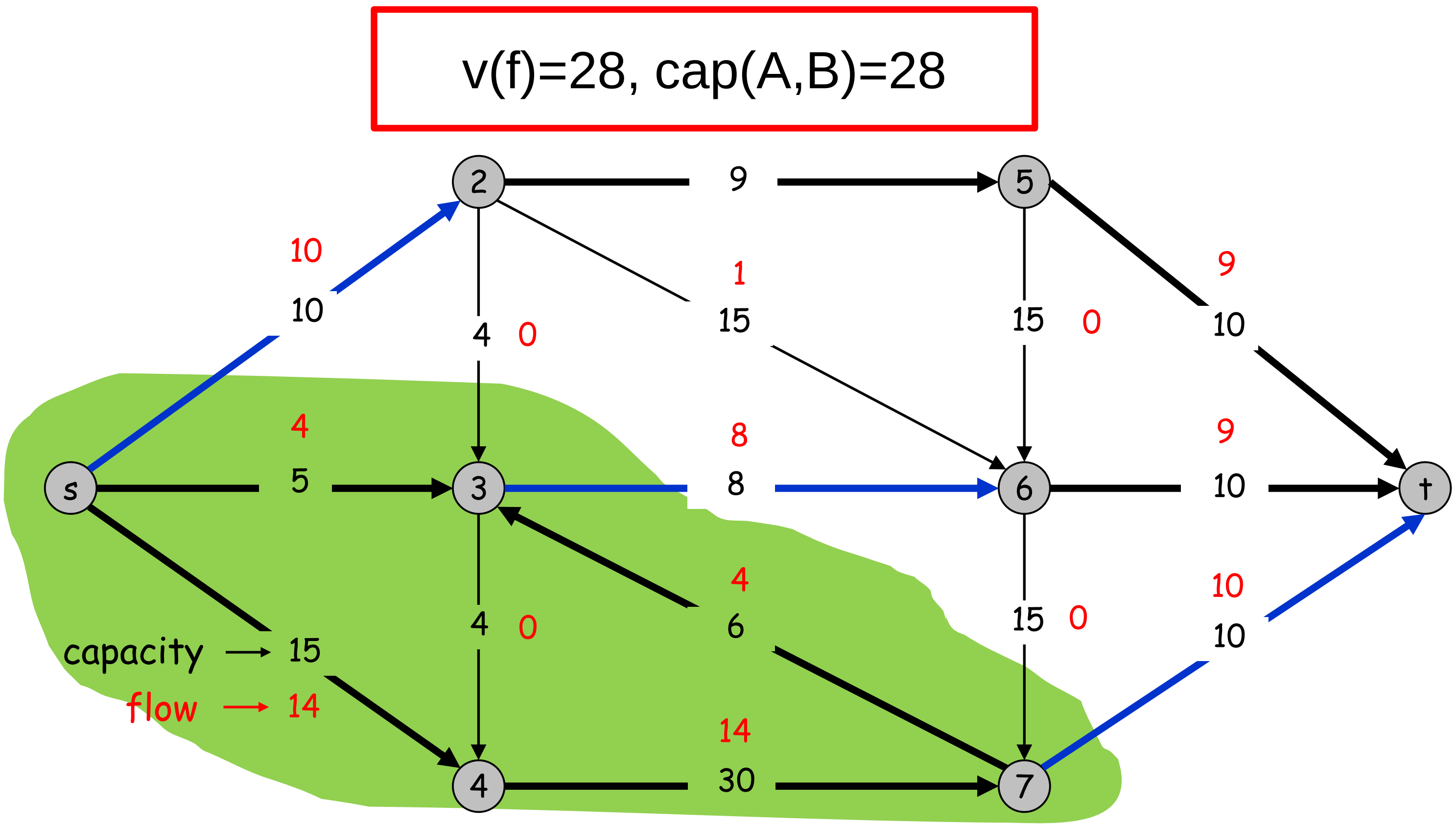


Certificate of Optimality

Corollary: Suppose there is a s-t cut (A,B) such that

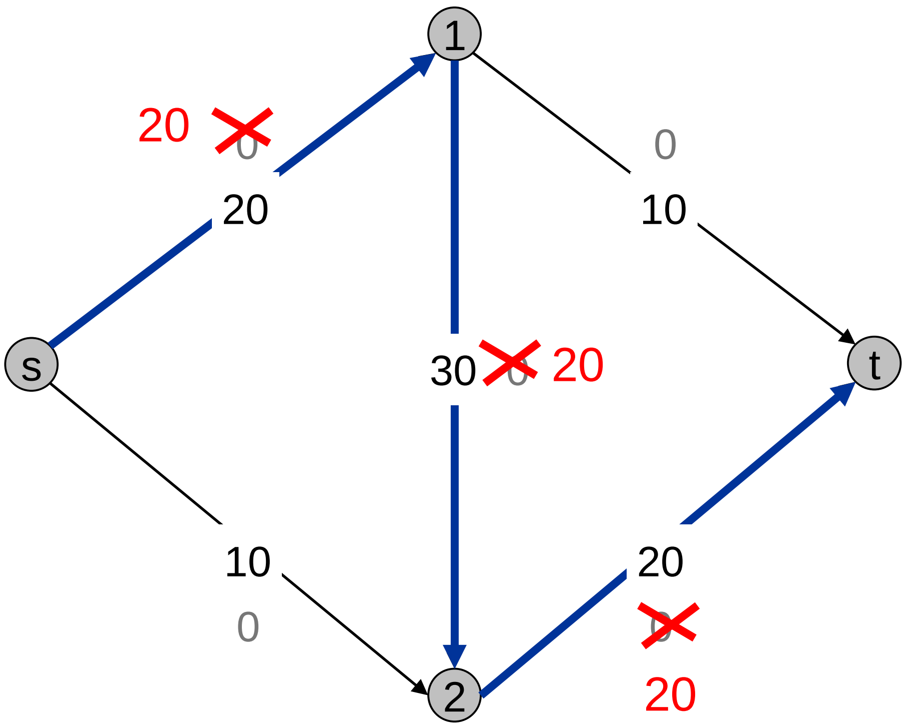
$$v(f) = cap(A, B)$$

Then, f is a maximum flow and (A,B) is a minimum cut.



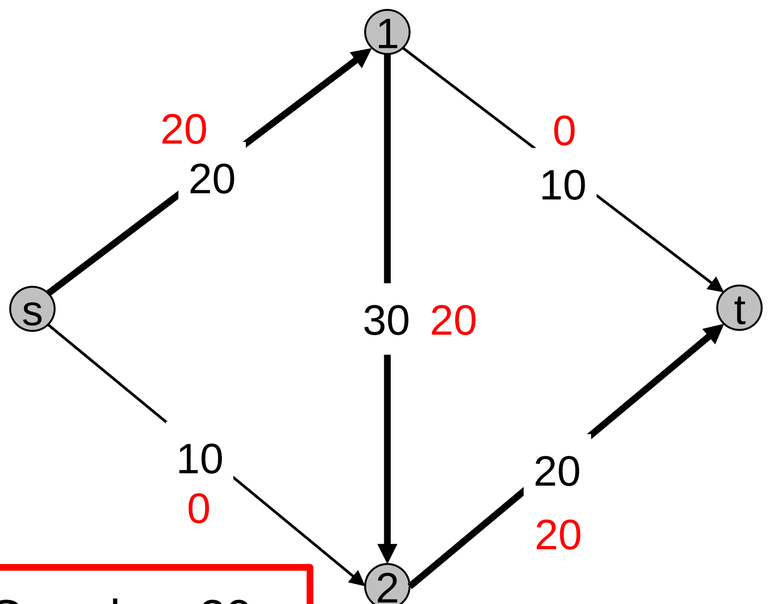
A Greedy Algorithm for Max Flow

- Start with f(e) = 0 for all edge e $\in$ E.
- Find an s-t path P where each edge has f(e) < c(e).
- Augment flow along path P.
- Repeat until you get stuck.

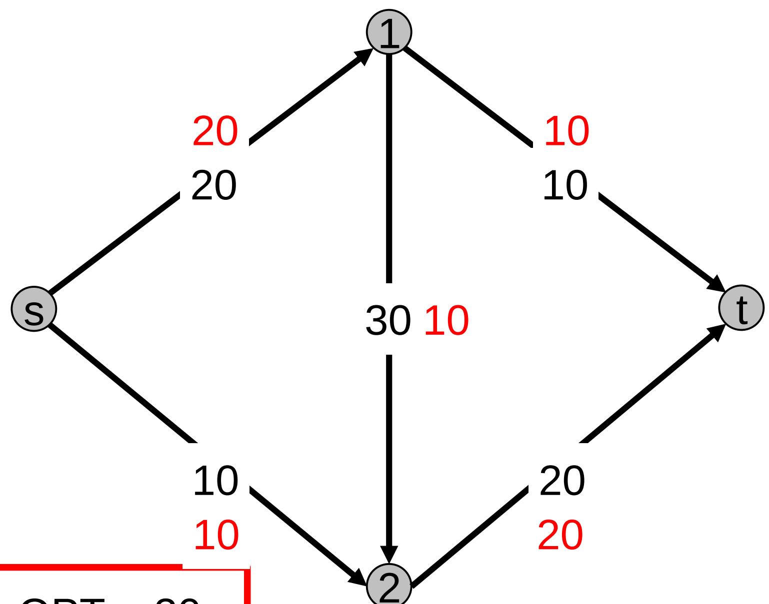


A Greedy Algorithm for Max Flow

- Start with f(e) = 0 for all edge e $\in$ E.
- Find an s-t path P where each edge has f(e) < c(e).
- Augment flow along path P.
- Repeat until you get stuck.

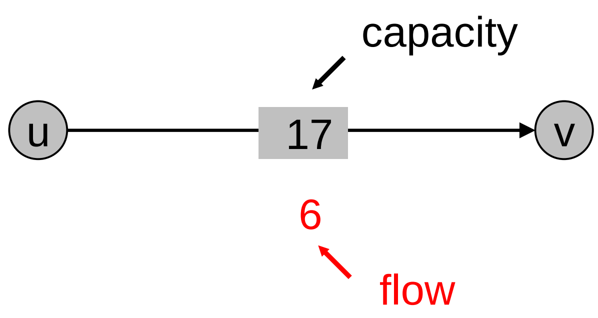


Greedy = 20



OPT = 30

Residual Graph



Original edge: e = (u, v) $\in$ E.

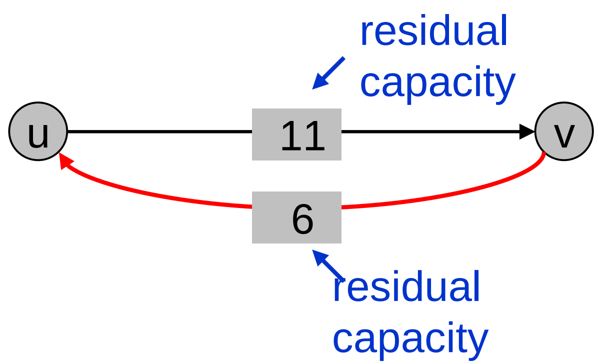
- Flow f(e), capacity c(e).

Residual edge.

- "Undo" flow sent.
- e = (u, v) and e^R = (v, u).

Residual capacity:

$$c_f(e) = \begin{cases} c(e) - f(e) & if\ e \in E \\ f(e) & if\ e^R \in E \end{cases}$$

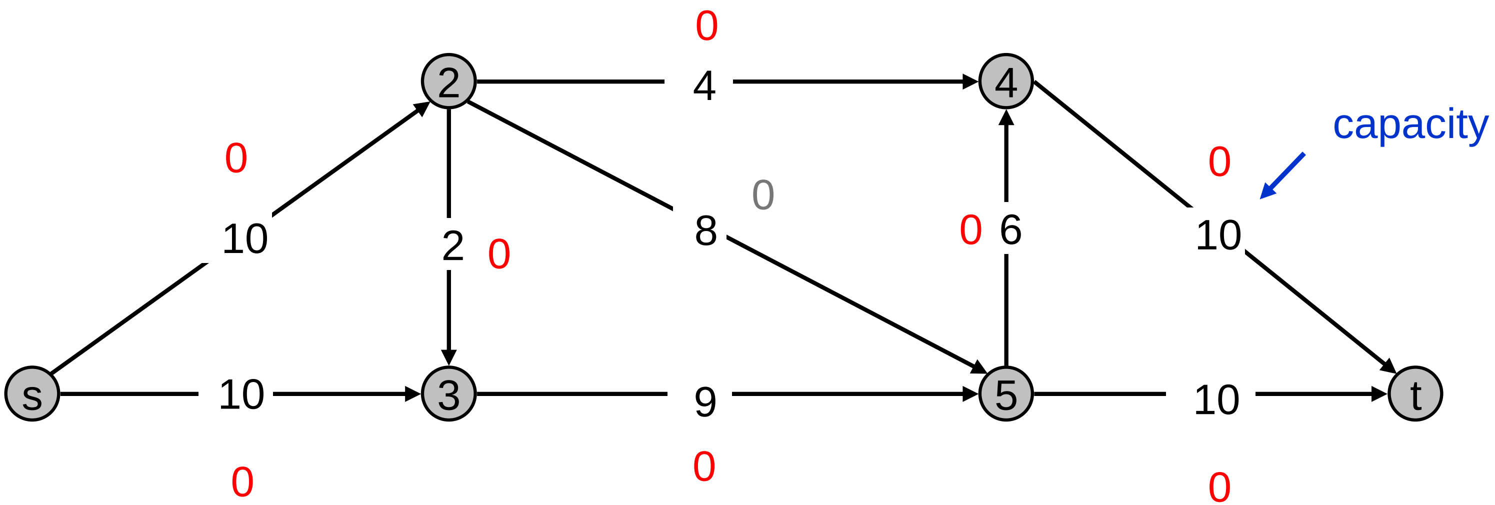


Residual graph: G_f = (V, E_f).

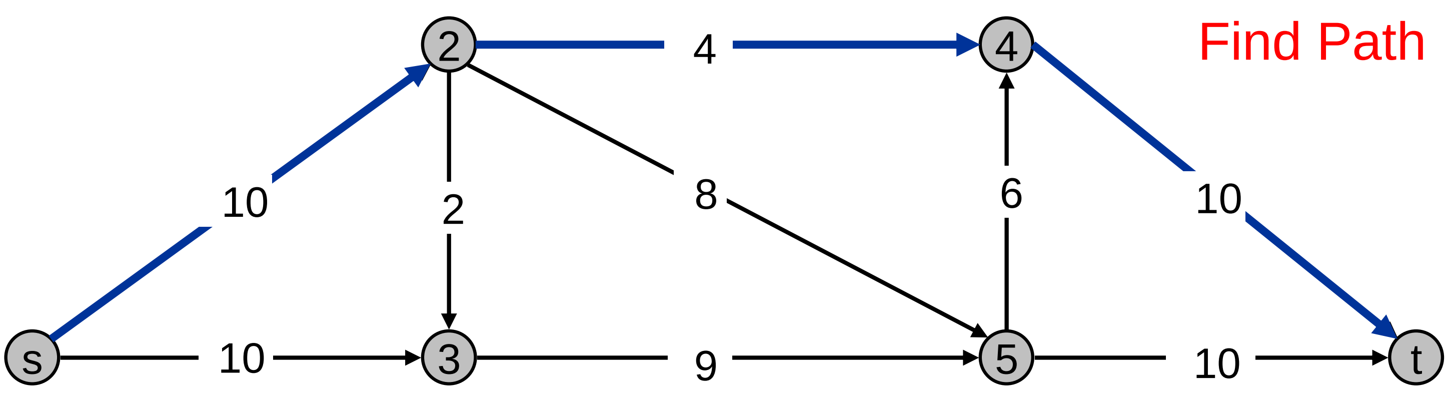
- Residual edges with positive residual capacity.
- $E_f \ = \ \{e : f(e) < \ c(e)\} \cup \{e^R \colon f(e) \ > \ 0\}$.

Ford-Fulkerson Alg: Greedy on G_f

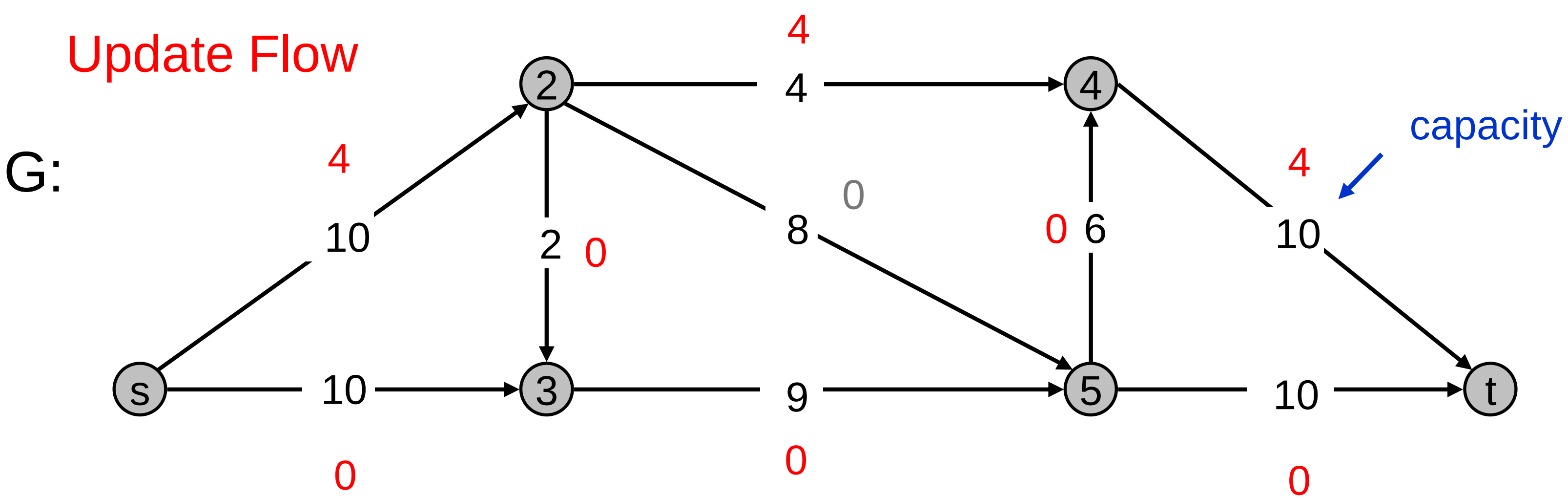
G:

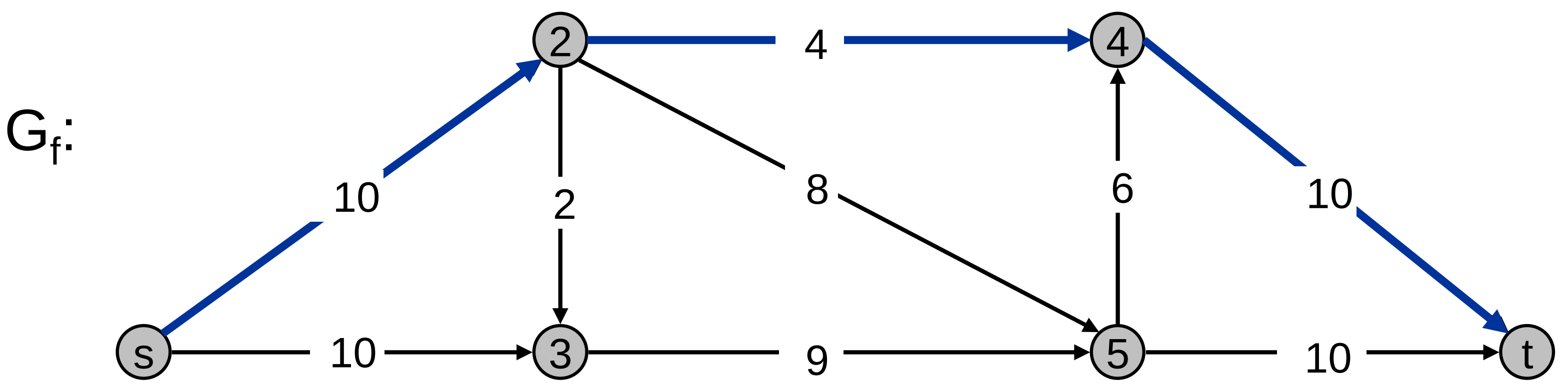


G_f:

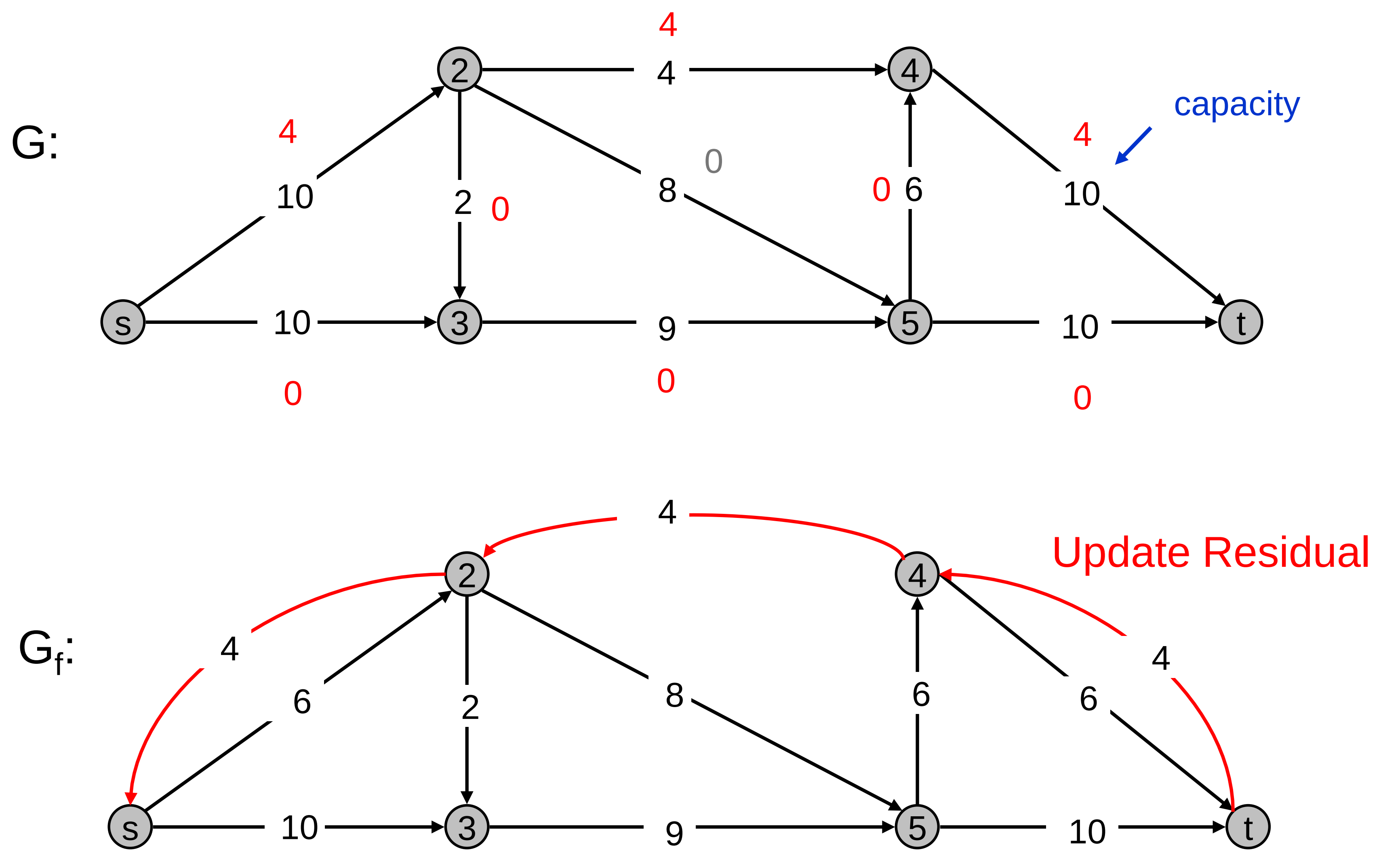


Ford-Fulkerson Alg: Greedy on G_f



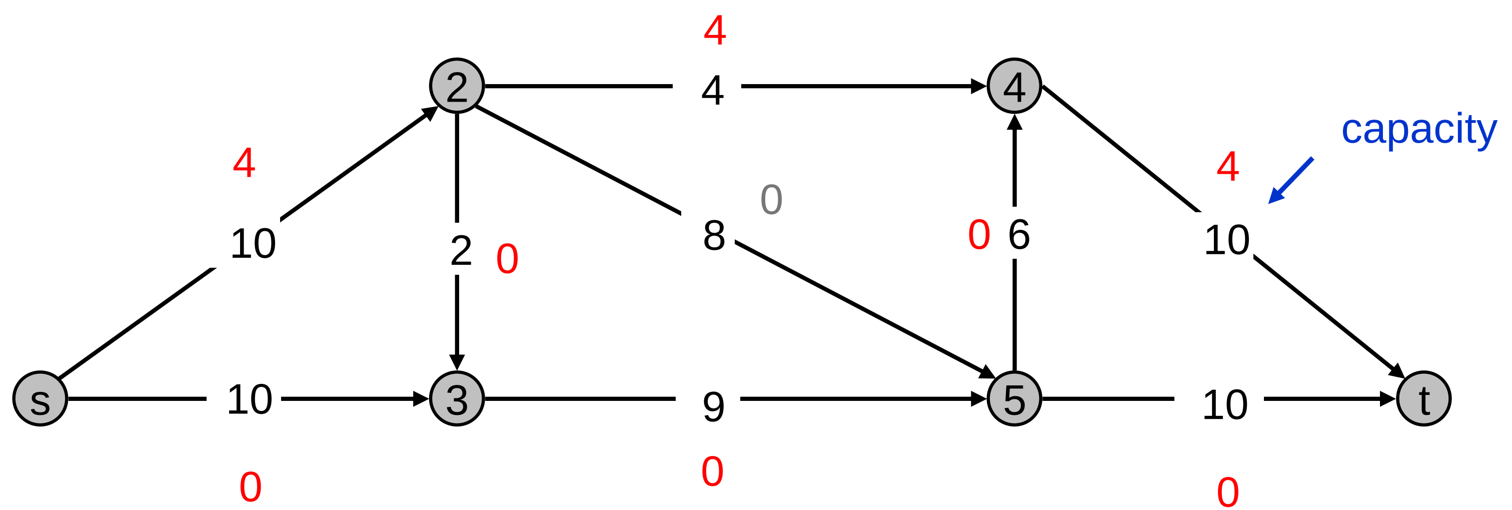


Ford-Fulkerson Alg: Greedy on G_f

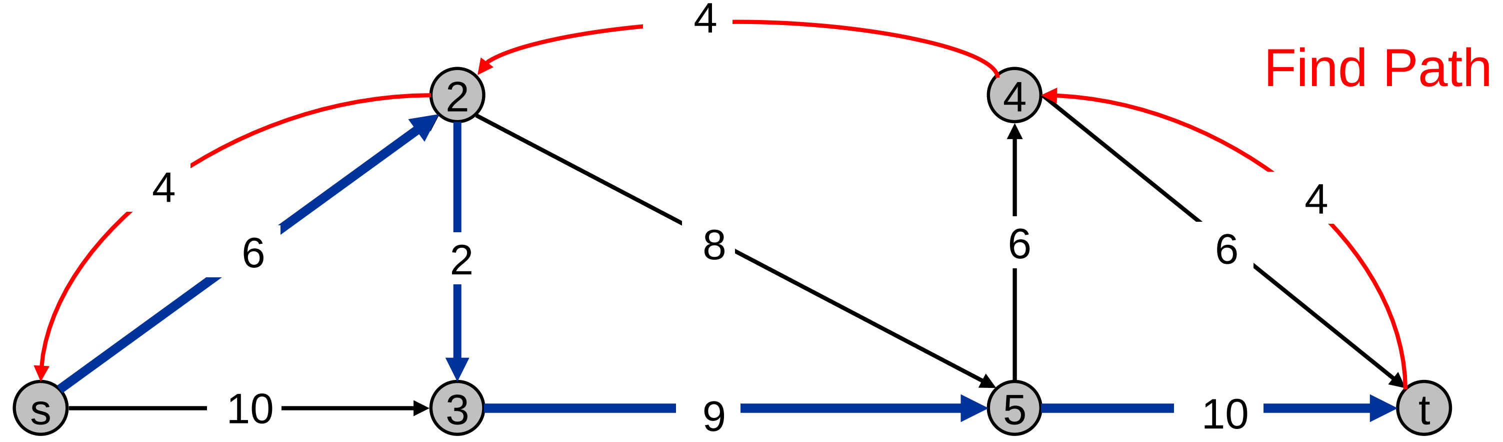


Ford-Fulkerson Alg: Greedy on G_f

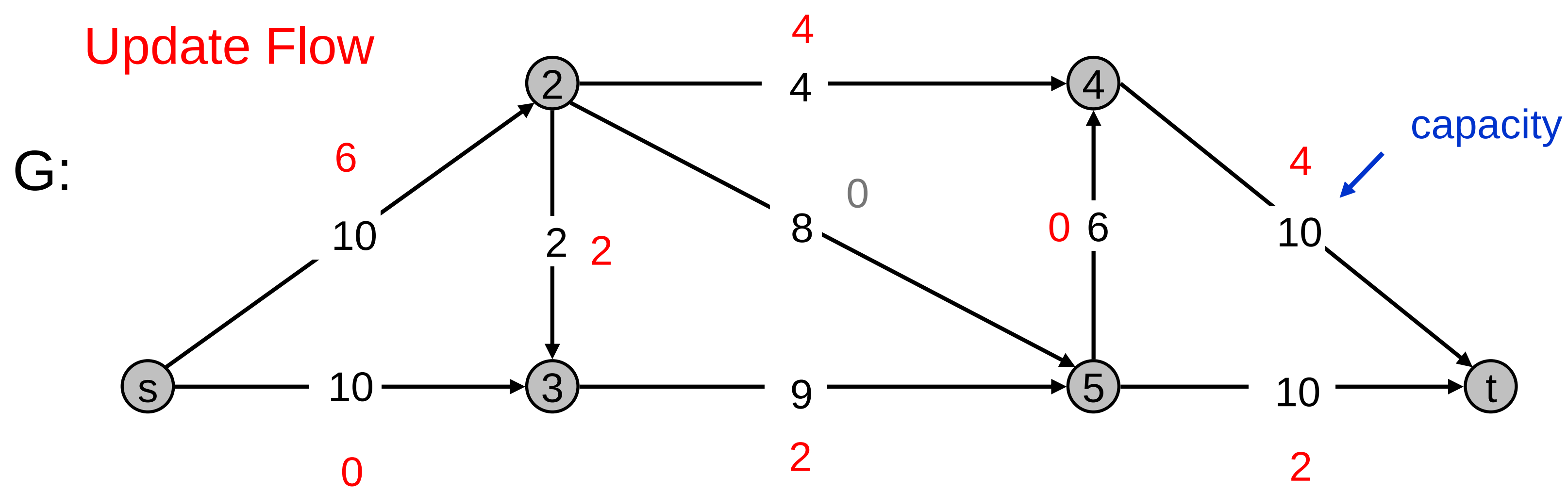
G:

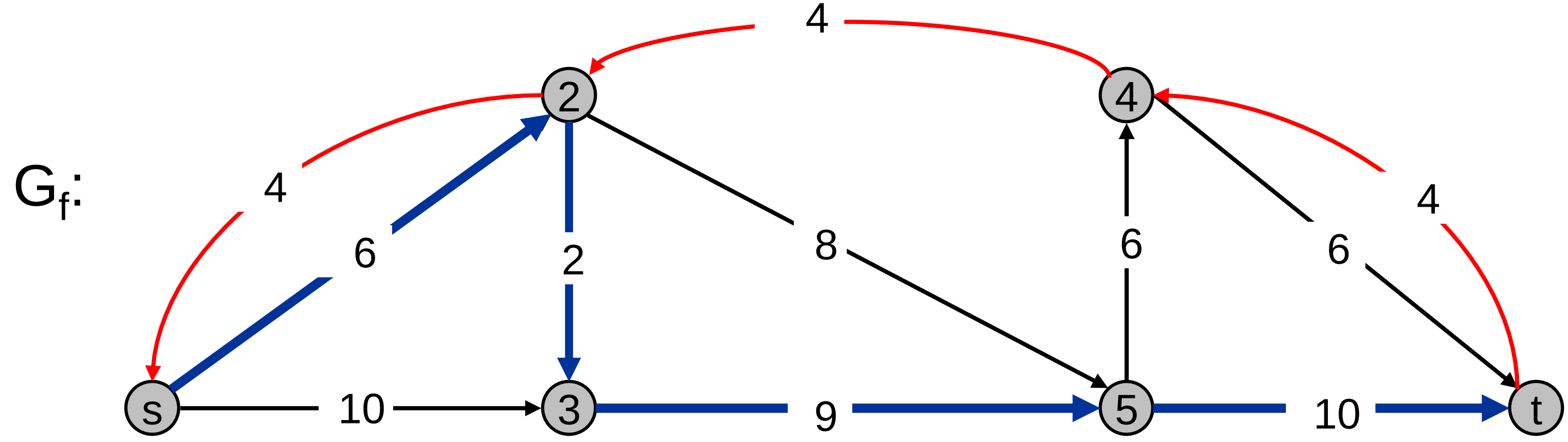


G_f:

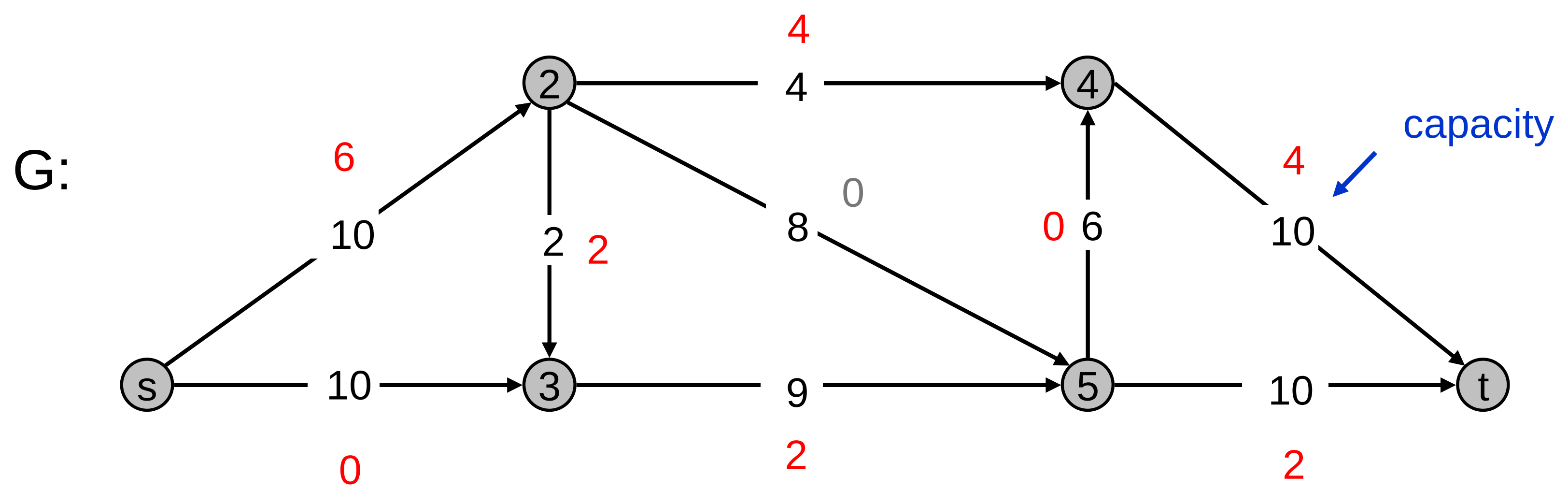


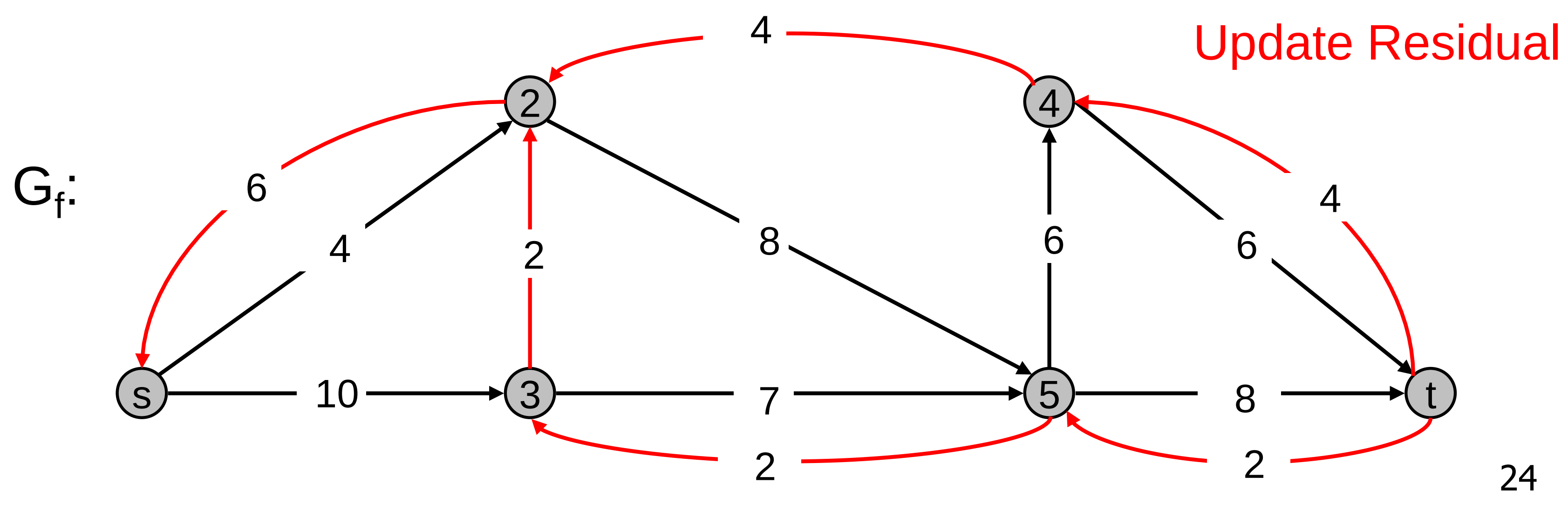
Ford-Fulkerson Alg: Greedy on G_f





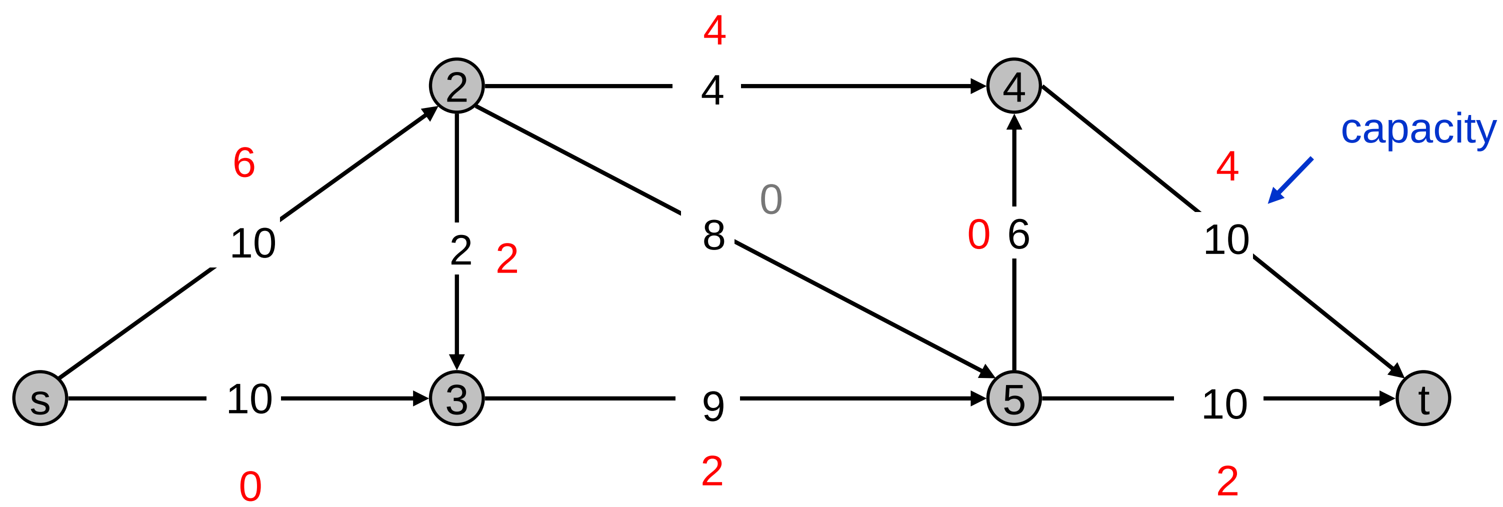
Ford-Fulkerson Alg: Greedy on G_f



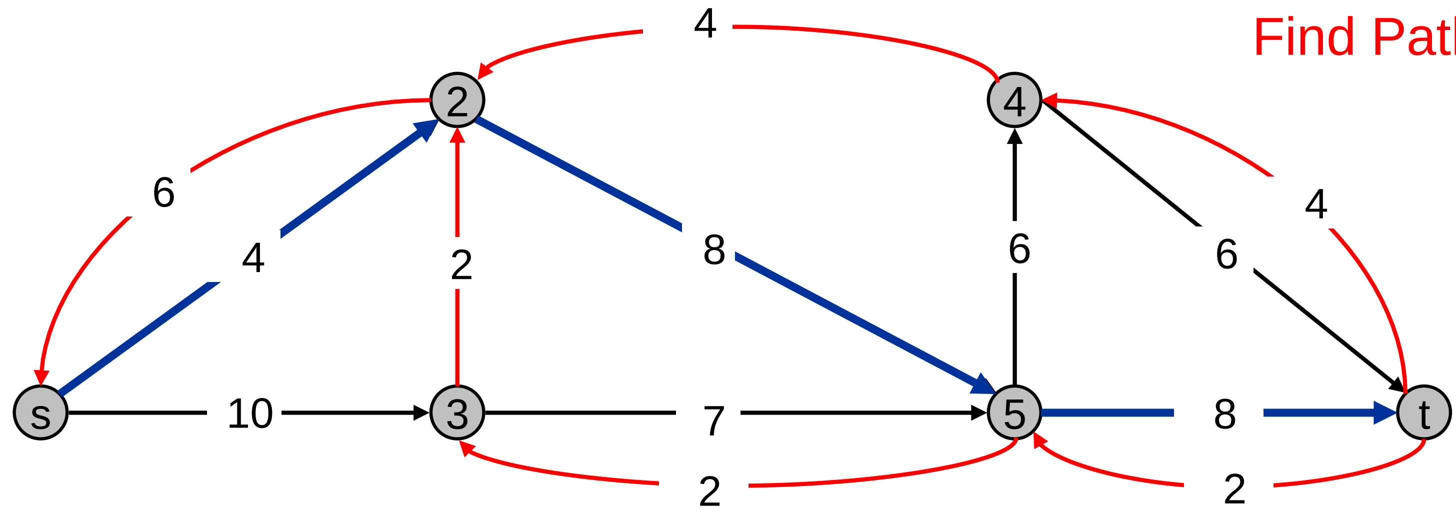


Ford-Fulkerson Alg: Greedy on G_f

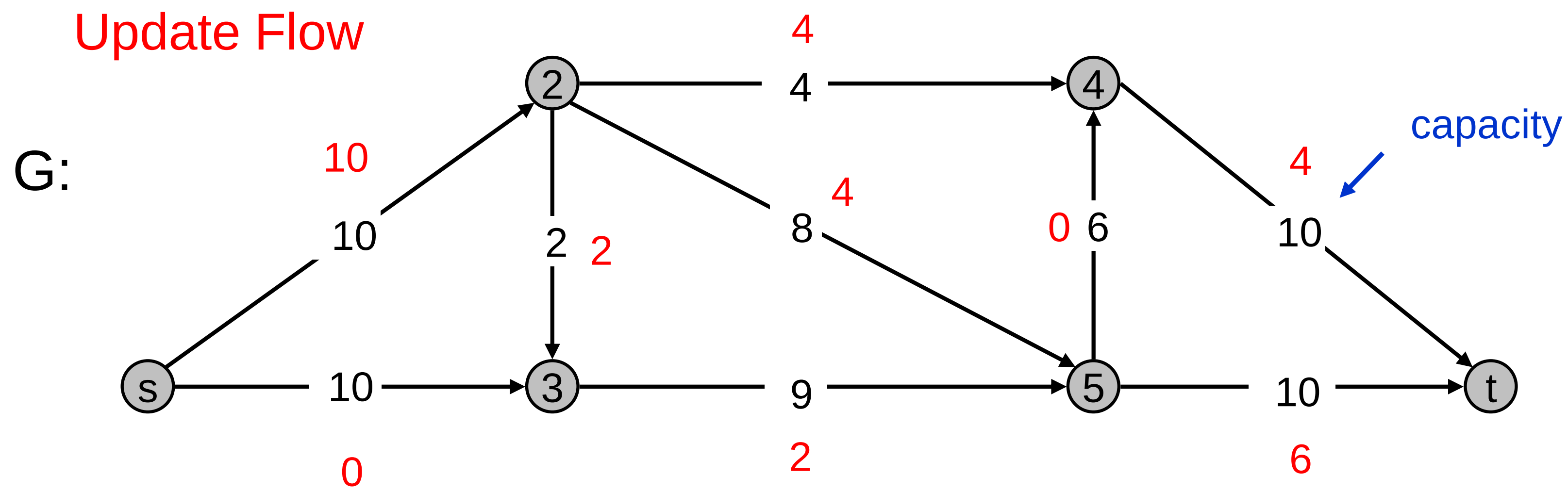
G:

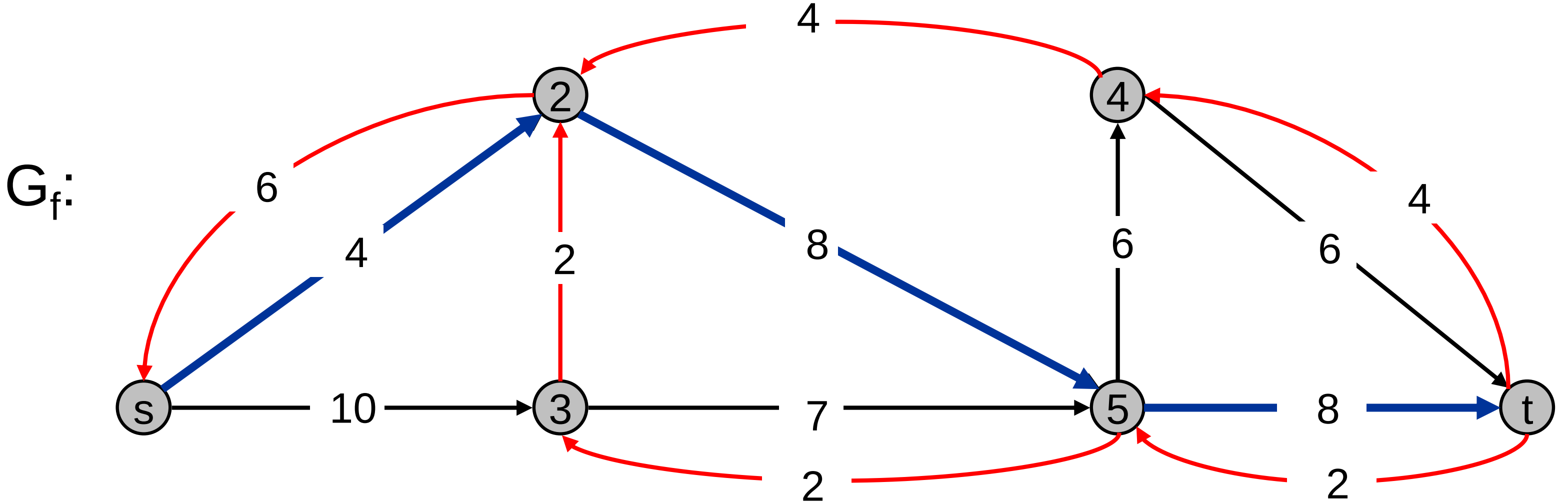


G_f:

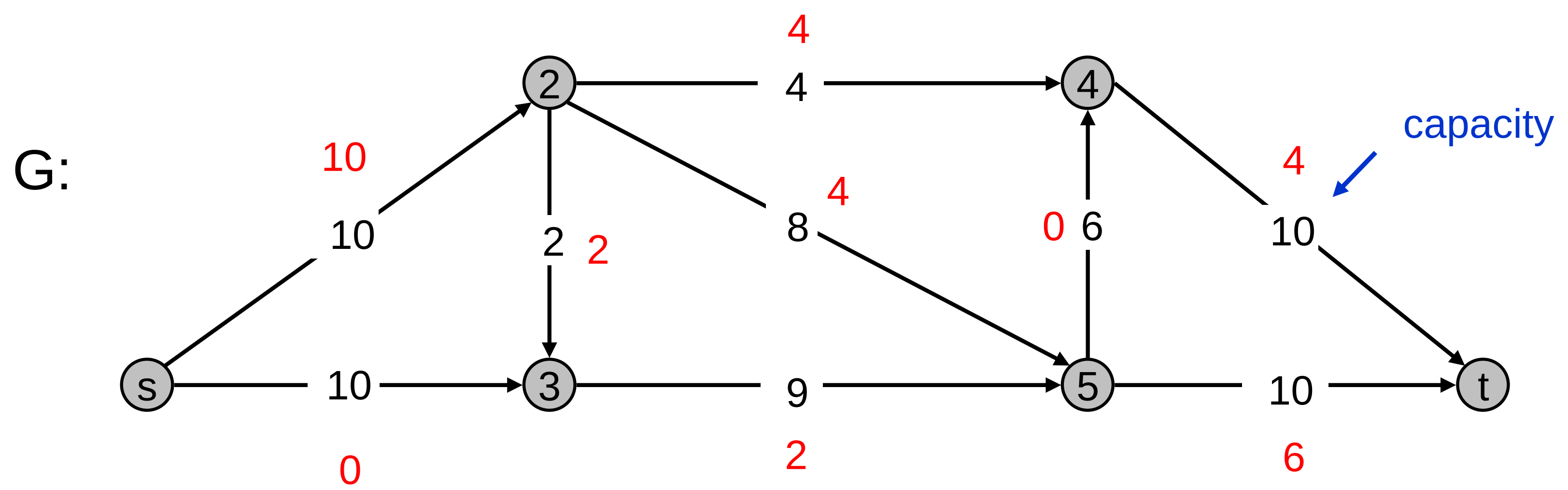


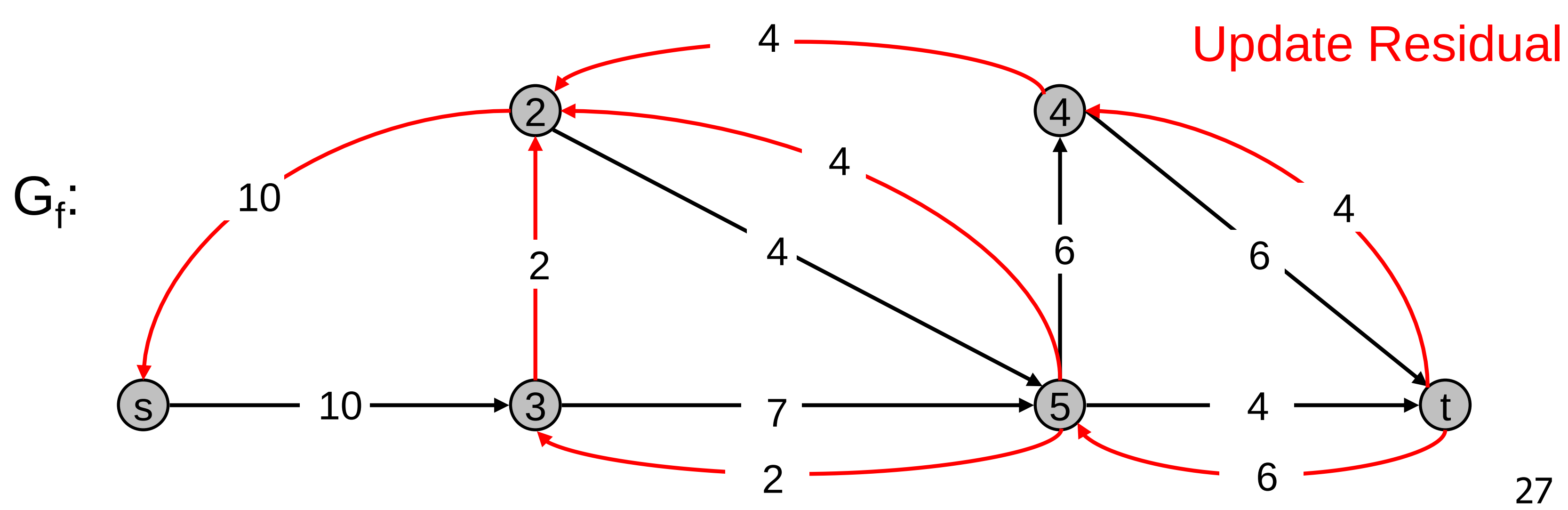
Ford-Fulkerson Alg: Greedy on G_f



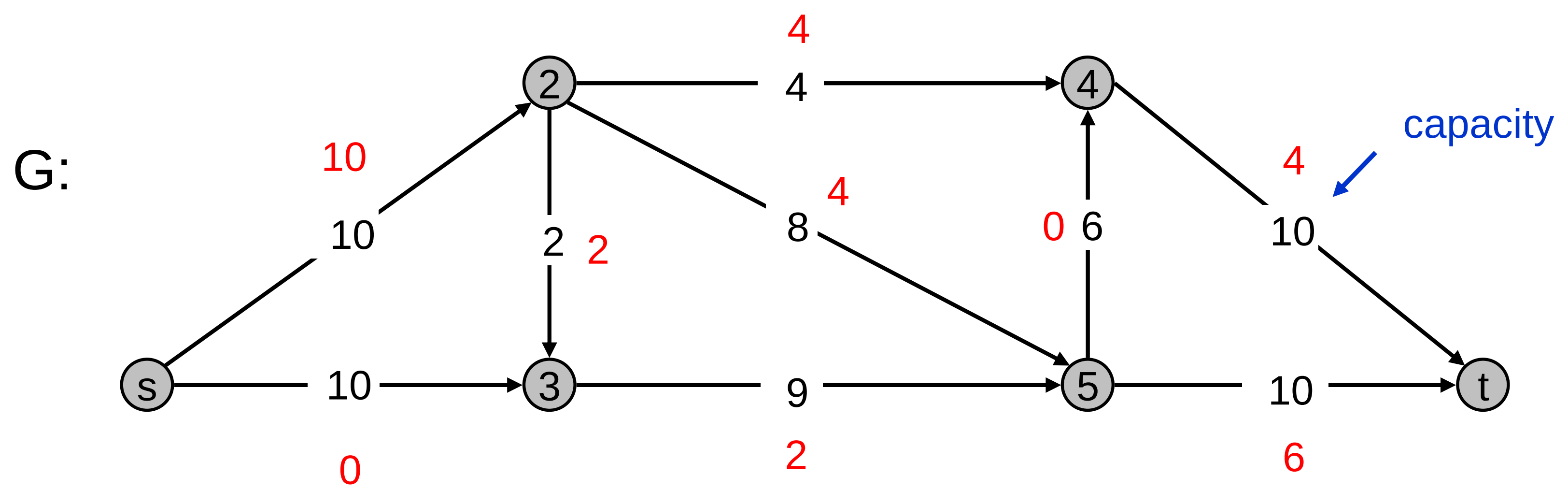


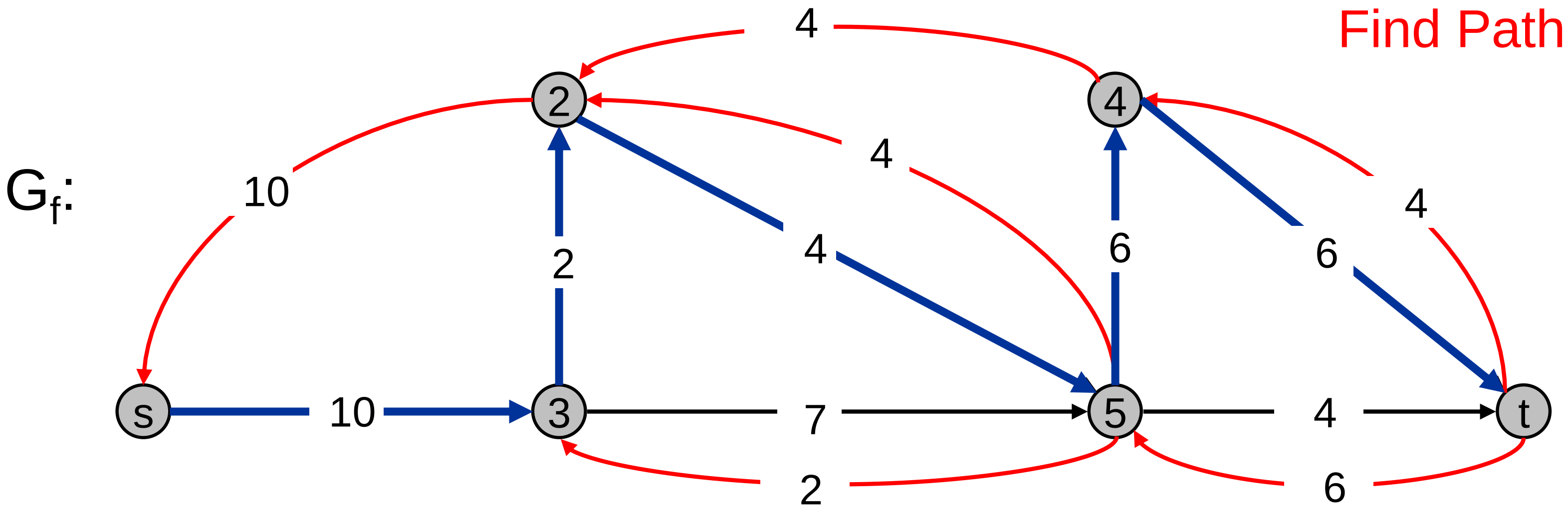
Ford-Fulkerson Alg: Greedy on G_f



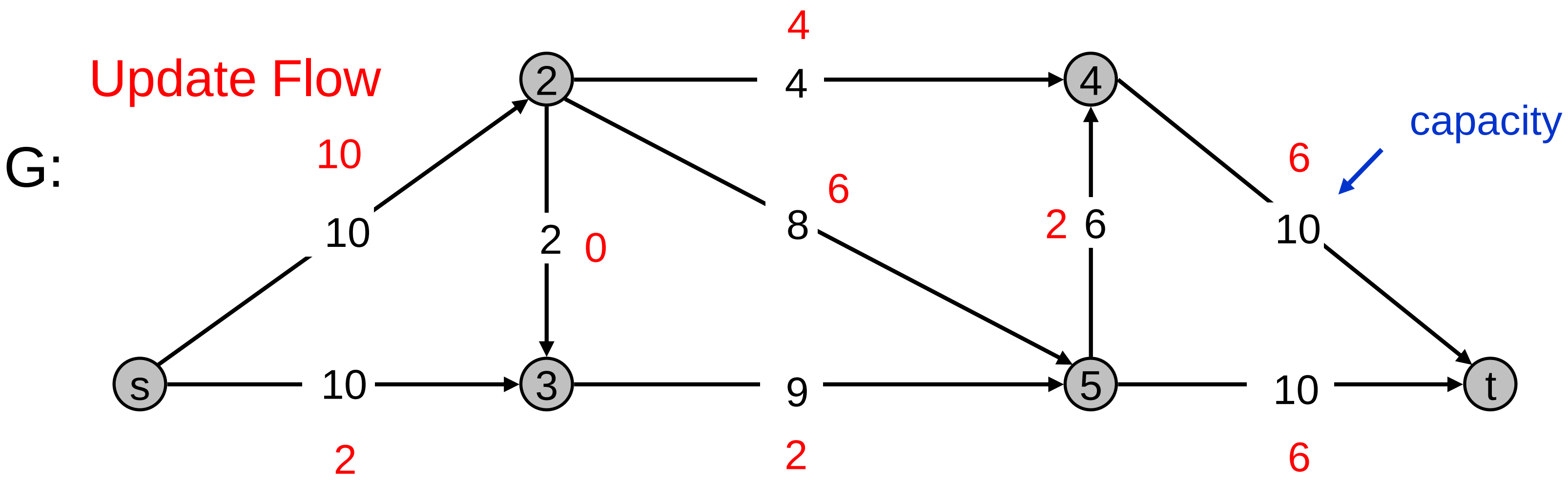


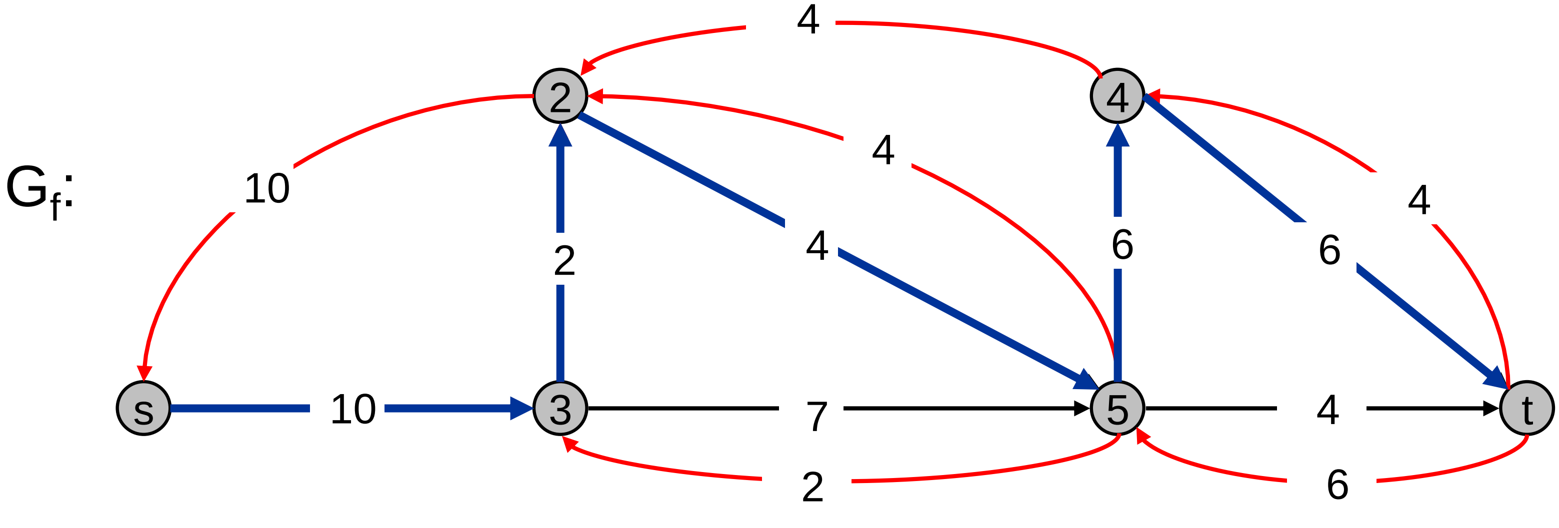
Ford-Fulkerson Alg: Greedy on G_f





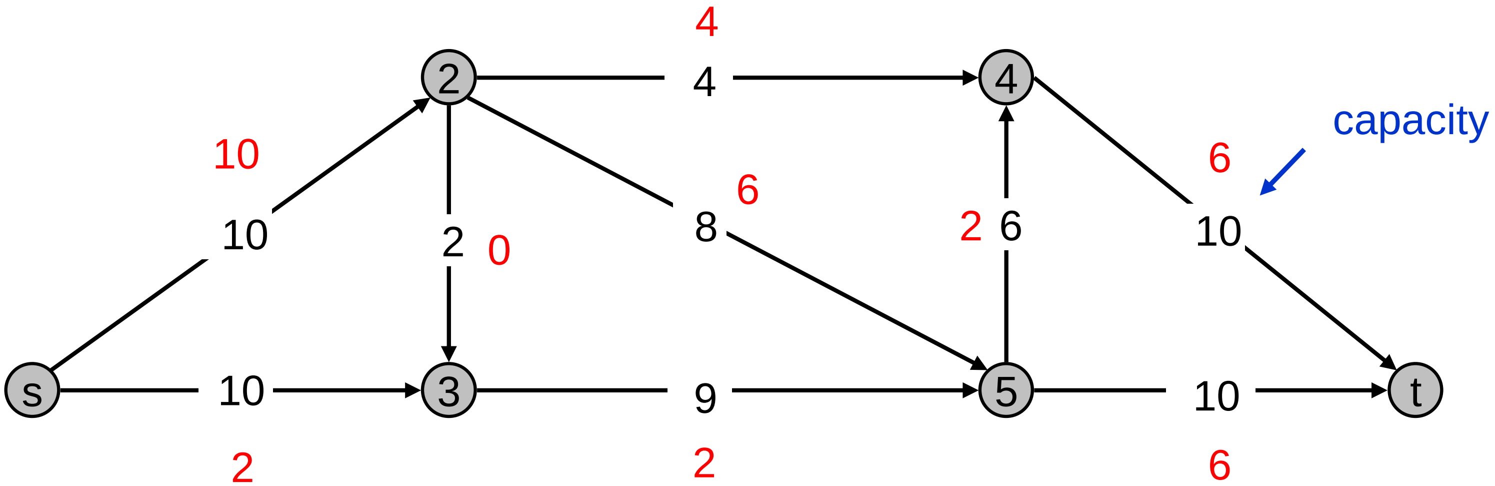
Ford-Fulkerson Alg: Greedy on G_f



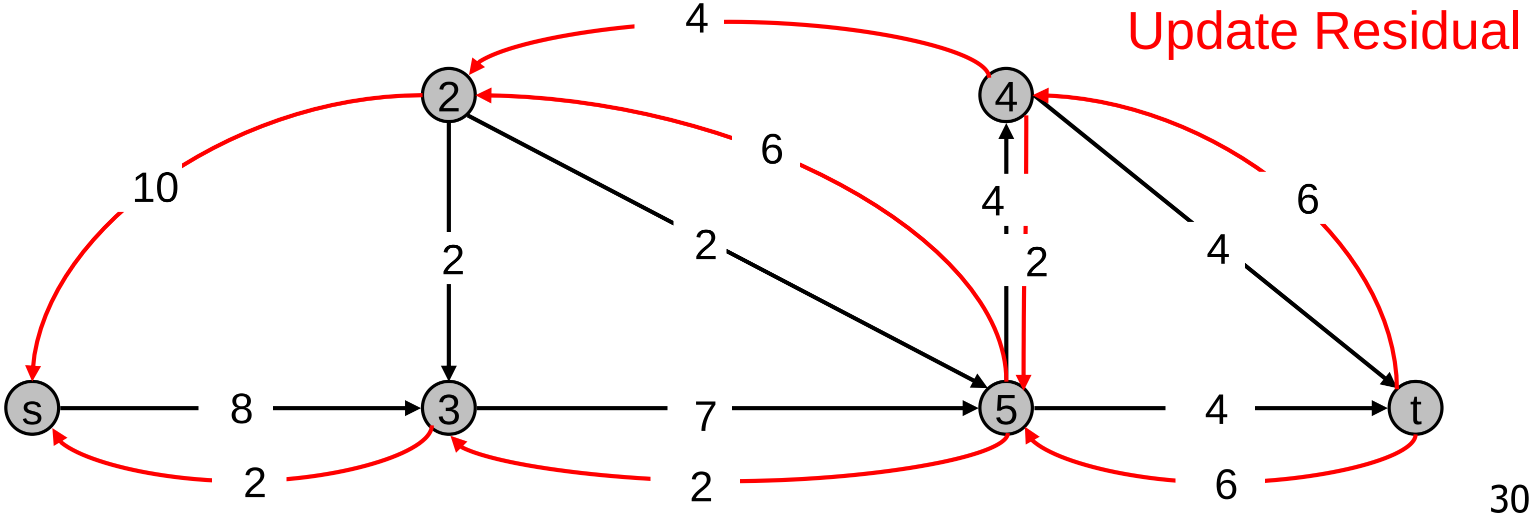


Ford-Fulkerson Alg: Greedy on G_f

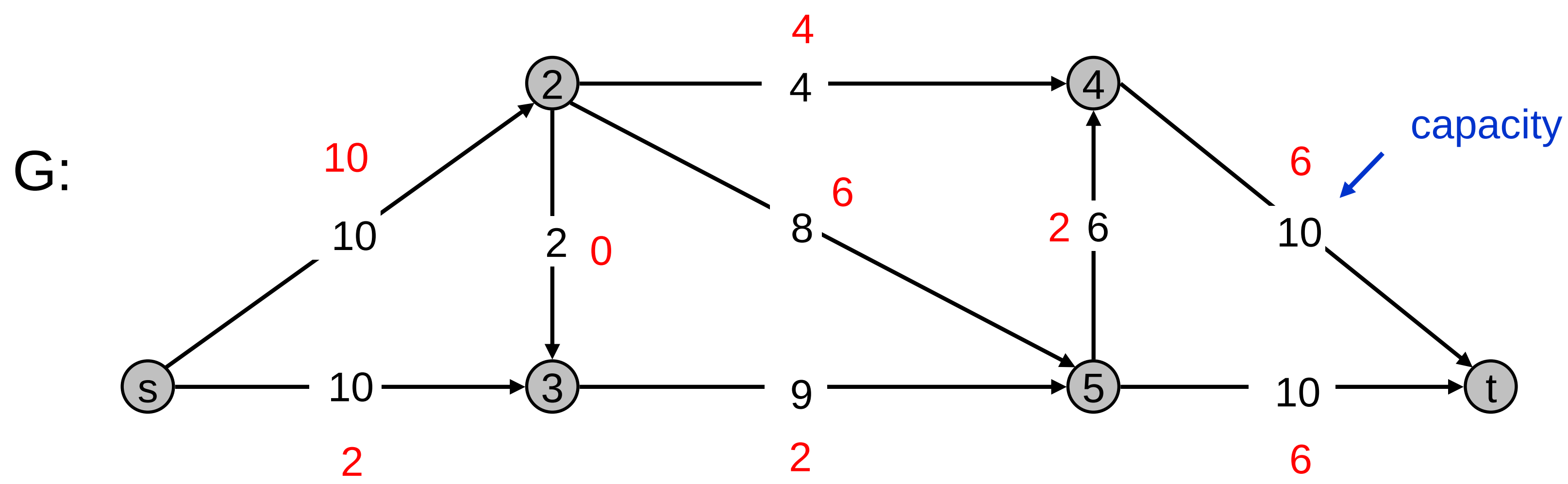
G:

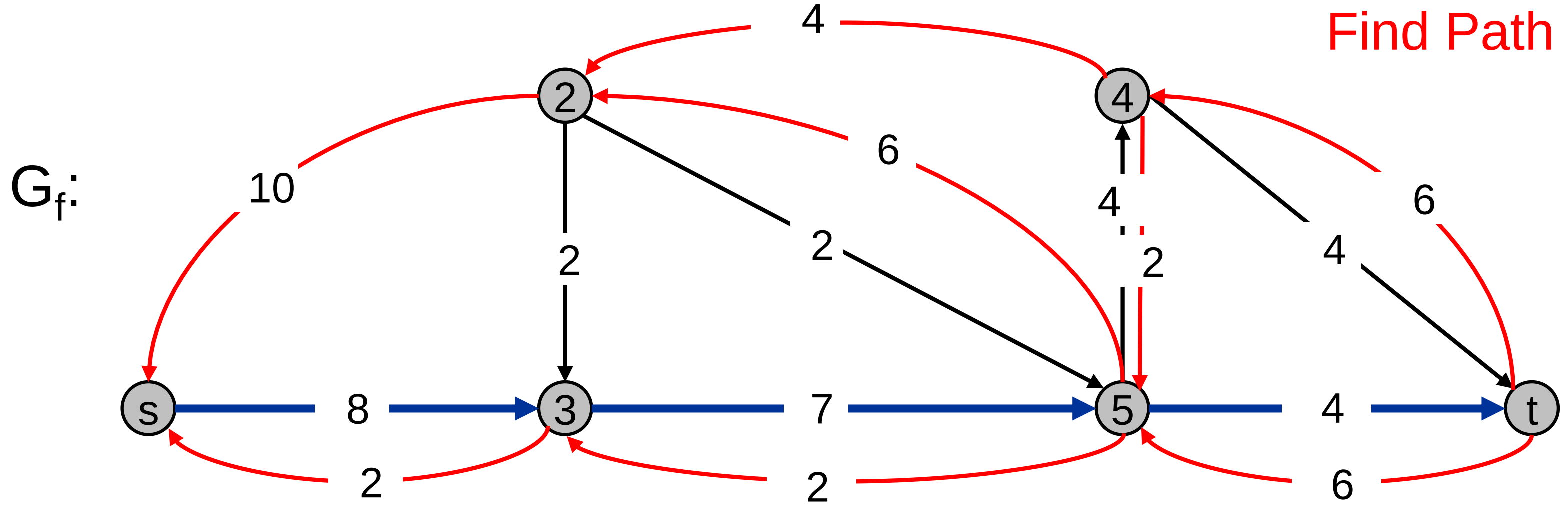


G_f:



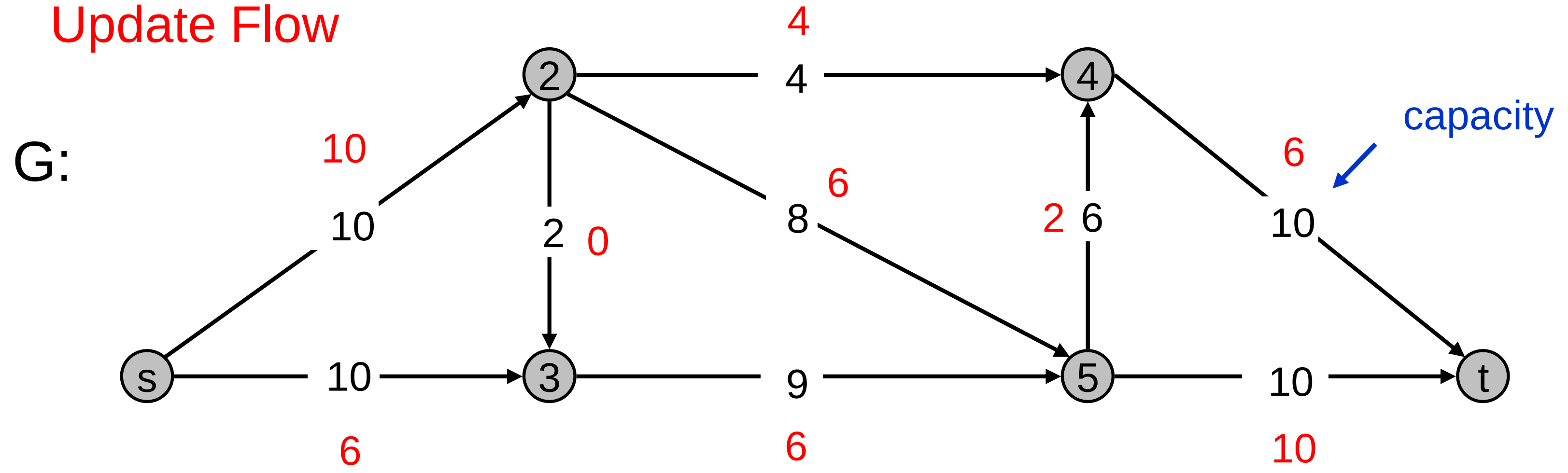
Ford-Fulkerson Alg: Greedy on G_f

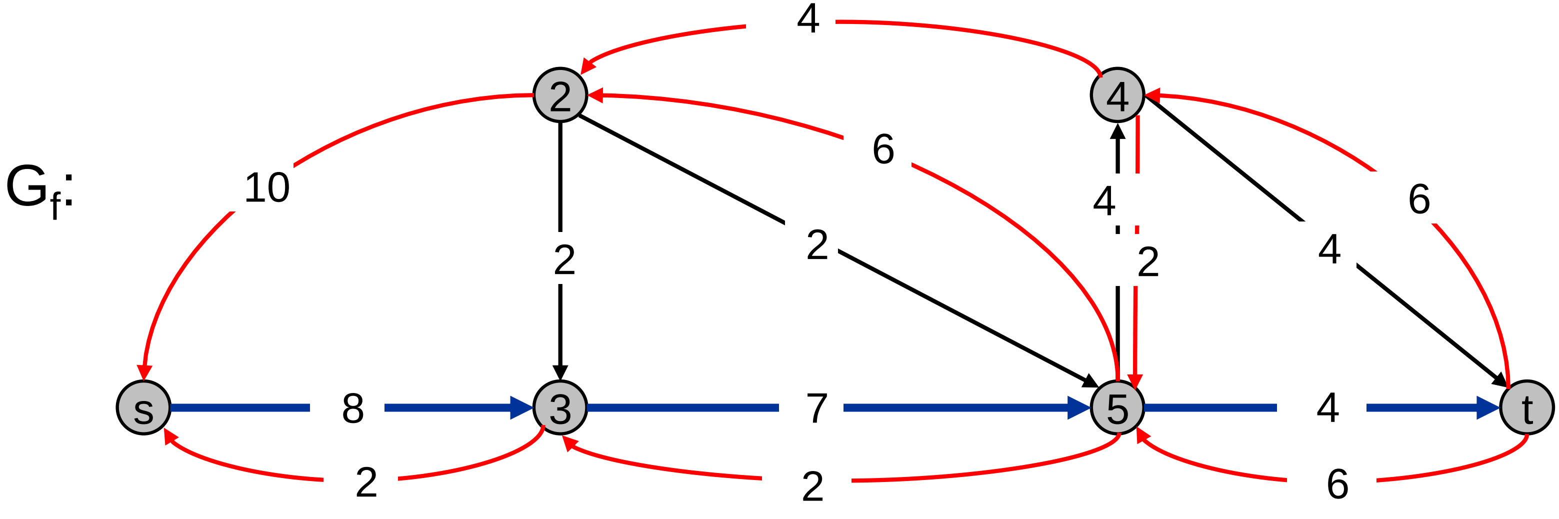




Ford-Fulkerson Alg: Greedy on G_f

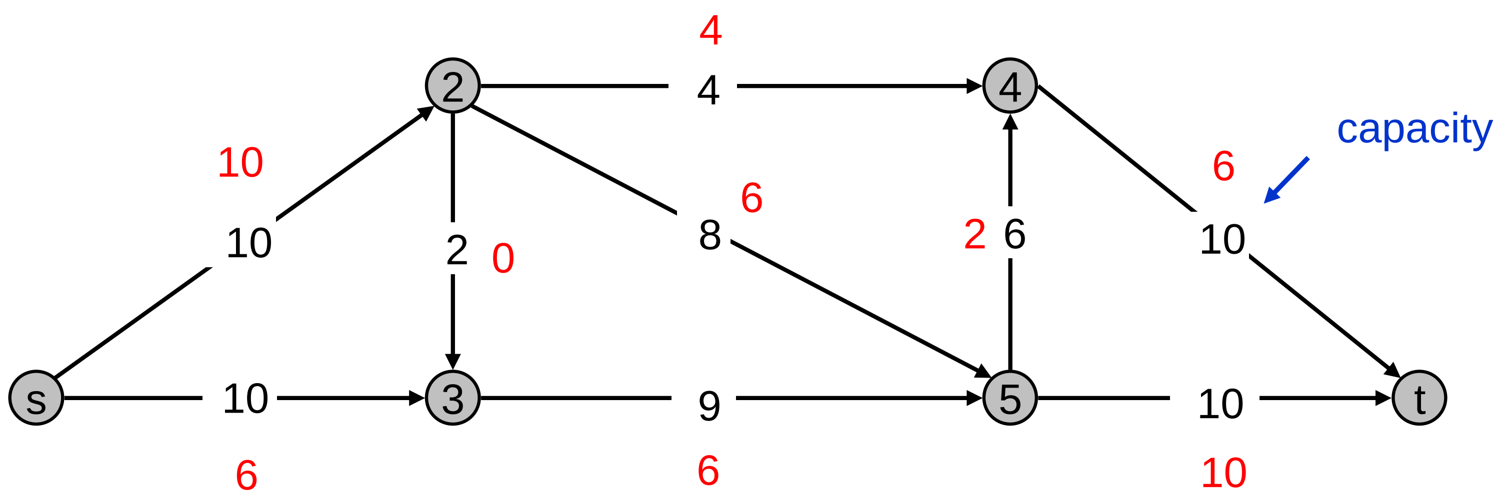
Update Flow



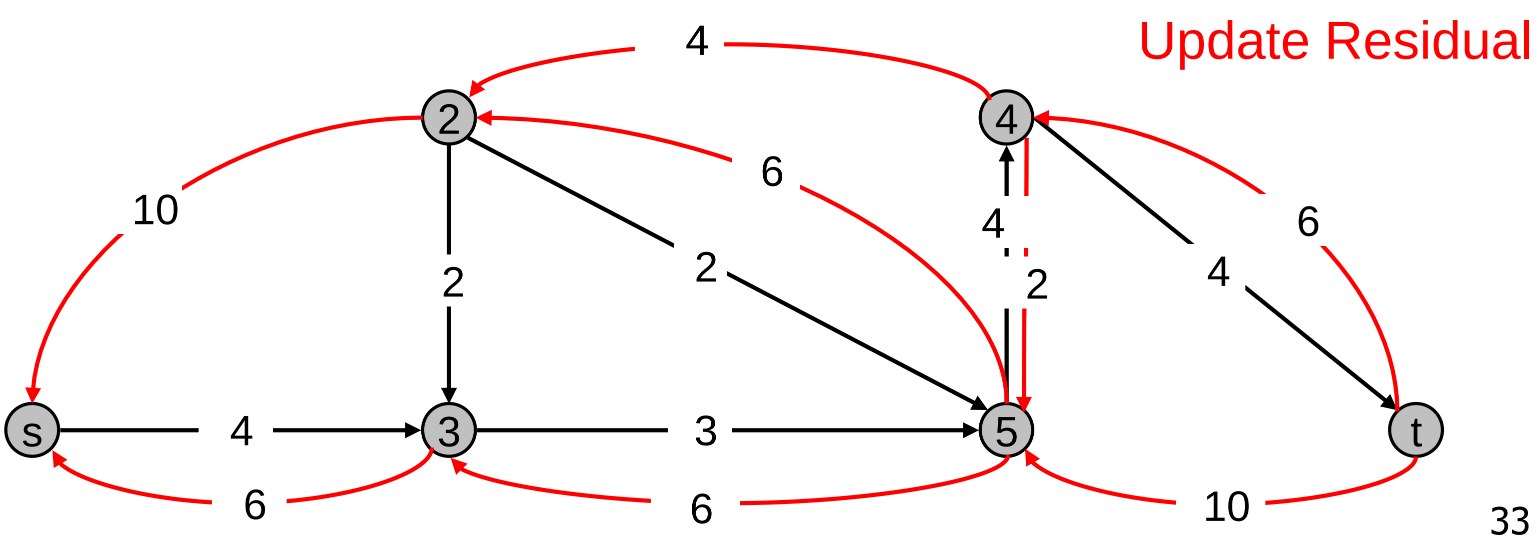


Ford-Fulkerson Alg: Greedy on G_f

G:

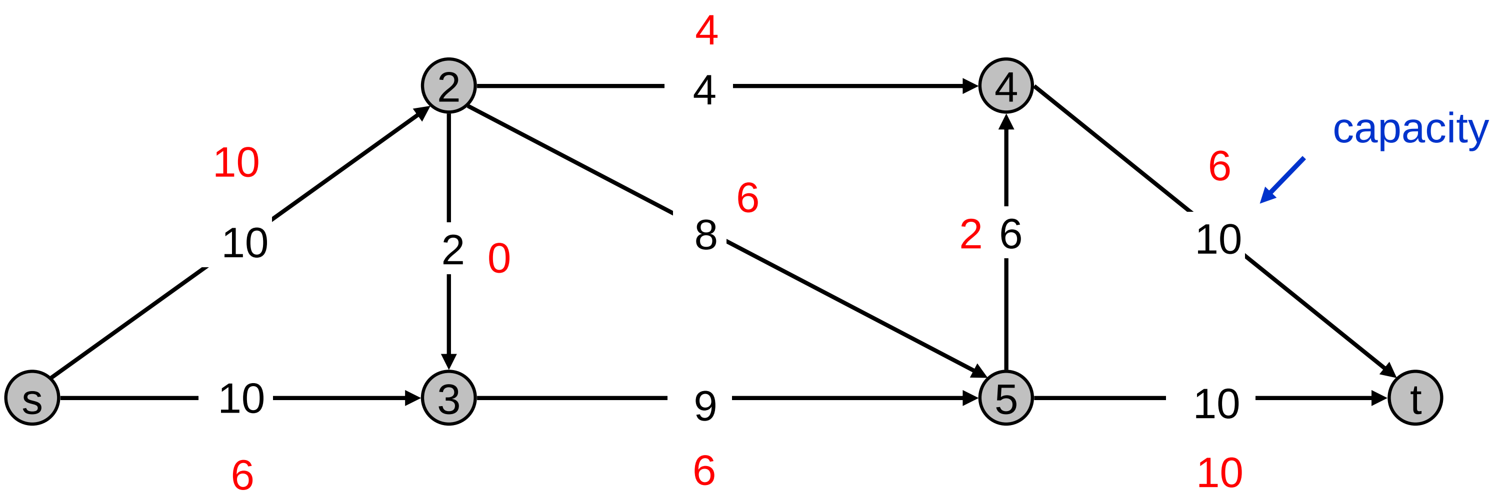


G_f:

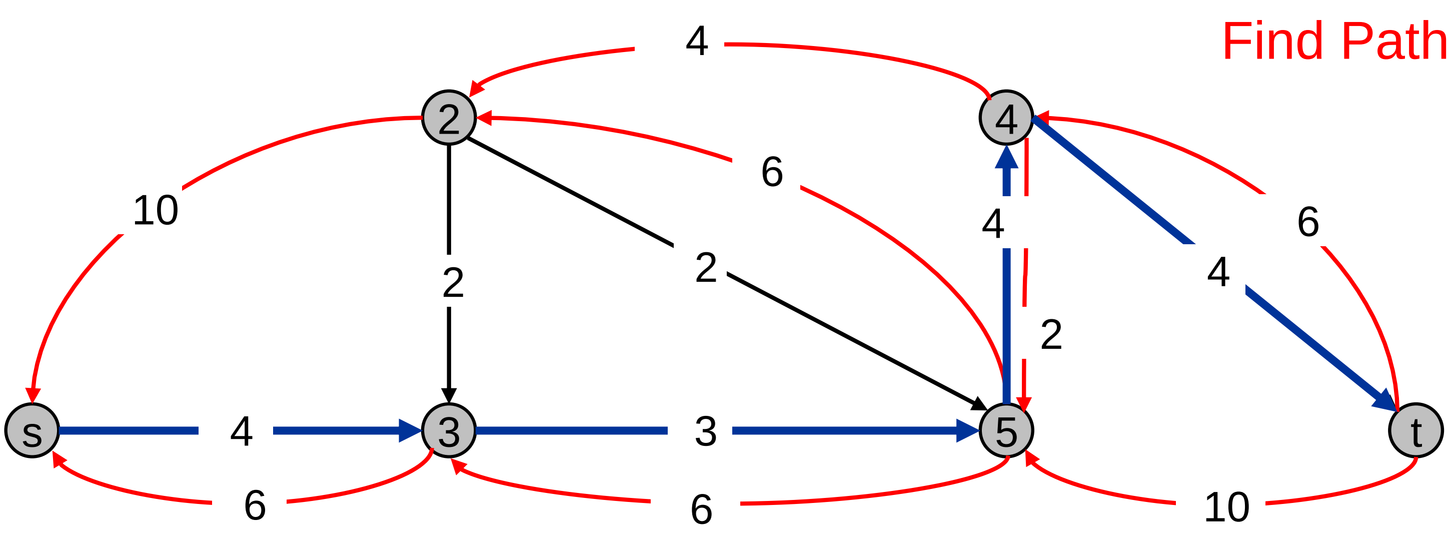


Ford-Fulkerson Alg: Greedy on G_f

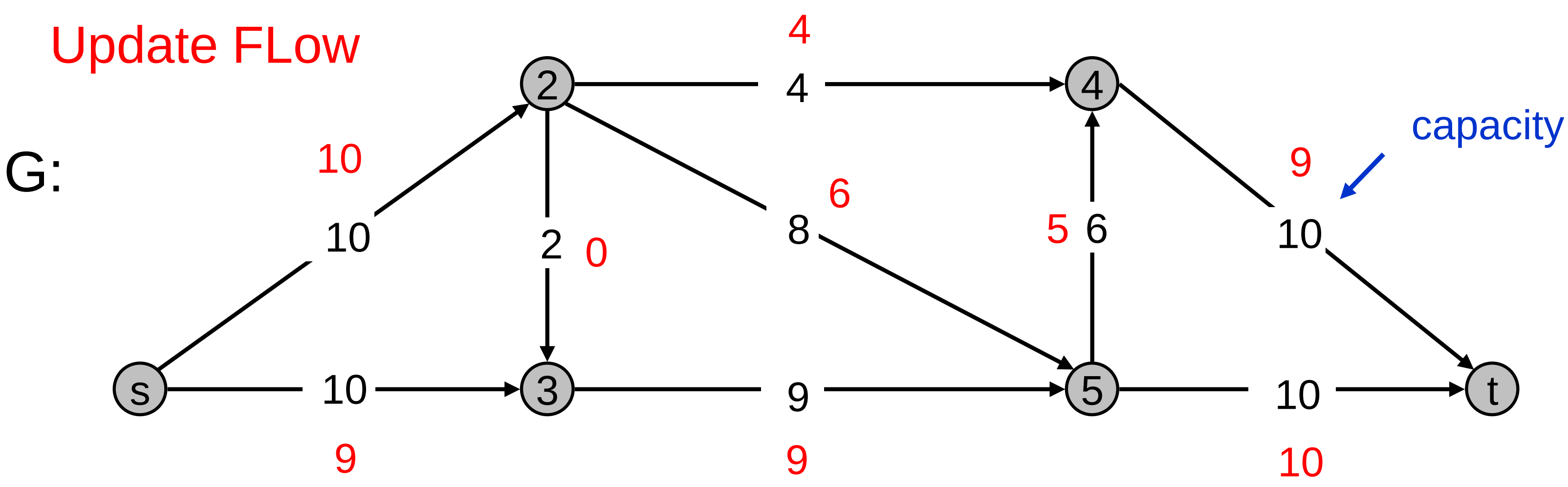
G:

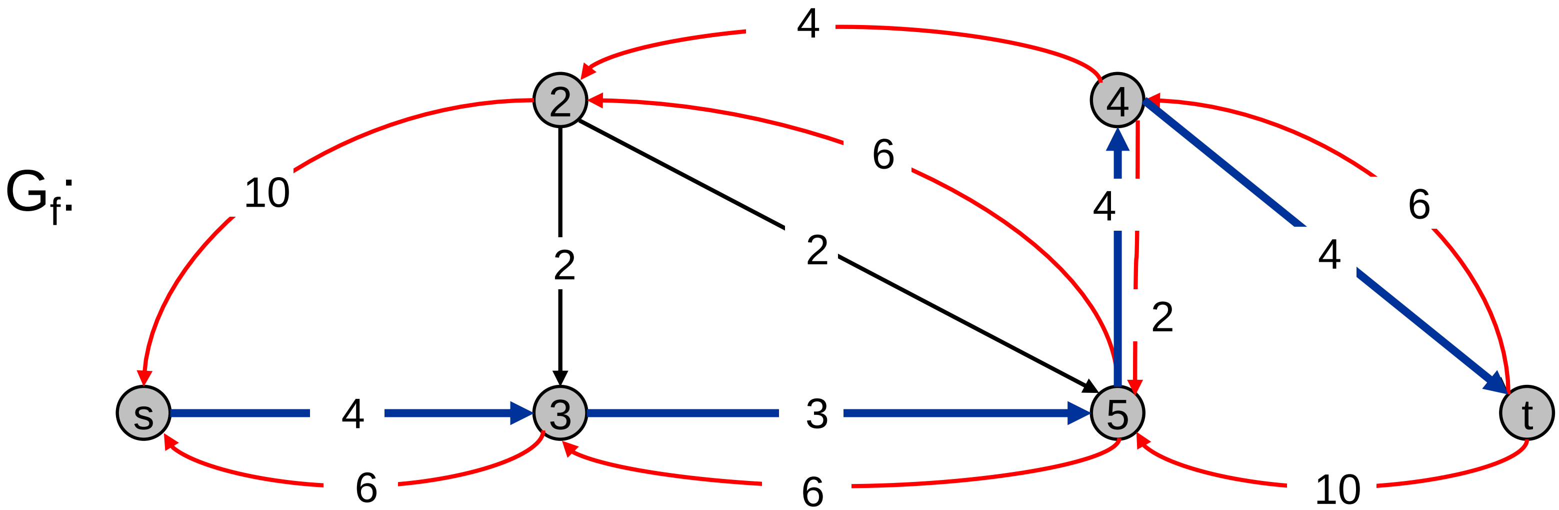


G_f:



Ford-Fulkerson Alg: Greedy on G_f





Ford-Fulkerson Alg: Greedy on G_f